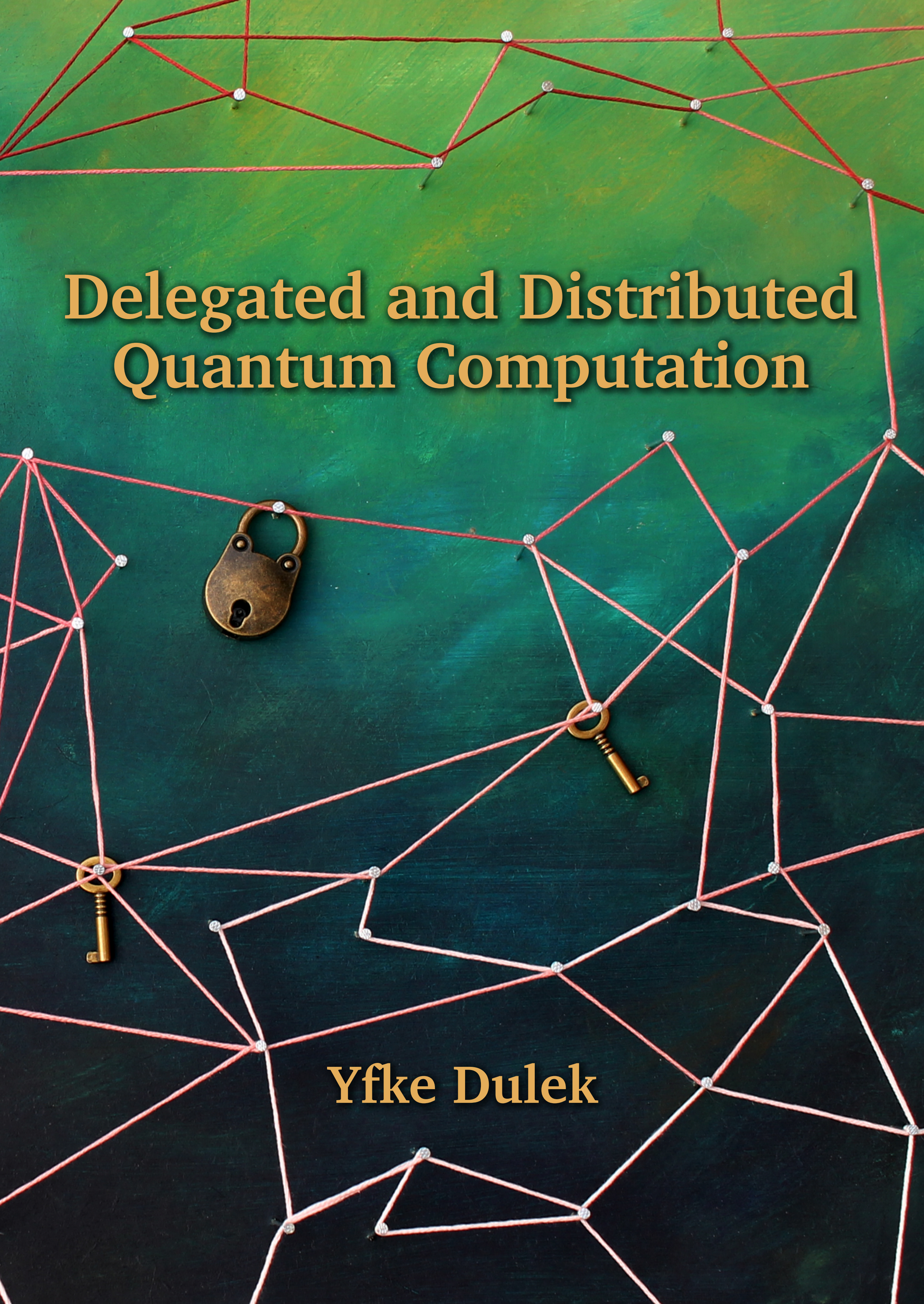


Delegated and Distributed Quantum Computation

Yfke Dulek

Delegated and Distributed Quantum Computation

ILLC Dissertation Series DS-2021-01

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: `illc@uva.nl`
homepage: `http://www.illc.uva.nl/`

Printed and bound by NBD Biblion.

ISBN: 978-90-9034003-6

Delegated and Distributed Quantum Computation

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 15 januari 2021, te 10.00 uur

door

Yfke Marie Dulek

geboren te Leiden

Promotiecommisie

Promotores

dr. C. Schaffner	Universiteit van Amsterdam
prof. dr. H.M. Buhrman	Universiteit van Amsterdam

Overige leden

dr. G. Alagic	University of Maryland
prof. dr. S.O. Fehr	Universiteit Leiden
dr. S.M. Jeffery	Centrum Wiskunde & Informatica
prof. dr. ir. C.T.A.M. de Laat	Universiteit van Amsterdam
prof. dr. S.J.L. Smets	Universiteit van Amsterdam
prof. dr. R.M. de Wolf	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica *voor Hein,*

die allang wist dat dit boekje vandaag in jouw handen zou liggen.

Contents

List of publications

The content of this dissertation is based on the following articles. The author names are ordered alphabetically, and all authors contributed equally.

[ADSS17] **Quantum fully homomorphic encryption with verification**
Gorjan Alagic, Yfke Dulek, Christian Schaffner, and Florian Speelman. In *Advances in Cryptology – ASIACRYPT 2017.*
Presented at QCrypt 2017, ASIACRYPT 2017, and QIP 2018.

[DS18] **Quantum ciphertext authentication and key recycling with the trap code**
Yfke Dulek and Florian Speelman. In *Proceedings of the 13th Conference on the Theory of Quantum Computation, Communication, and Cryptography (2018).*
Presented at TQC 2018.

[DGJ+20] **Secure multi-party quantum computation with a dishonest majority**
Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. In *Advances in Cryptology – EUROCRYPT 2020.*
Presented at EUROCRYPT 2020 and QCrypt 2020.

[ABDS20] **Impossibility of quantum virtual black-box obfuscation of classical circuits**
Gorjan Alagic, Zvika Brakerski, Yfke Dulek, and Christian Schaffner. *ArXiv preprint.*
Presented at QCrypt 2020.

Due to its close connection with [ADSS17], content from the author's MSc research is also included in this dissertation. (Parts of) Sections 5.1 to 5.3, 6.1 to 6.3 and 6.6 are based on this work:

[DSS16] **Quantum homomorphic encryption for polynomial-sized circuits**
Yfke Dulek, Christian Schaffner, and Florian Speelman. In *Theory of Computing, Vol 14 (2018).*
Presented at CRYPTO 2016, QCrypt 2016, ICITS 2016, and QIP 2017.

During the course of her PhD, the author has additionally co-authored the following work, which is not included in this dissertation:

[MDCA20] **Device-independent quantum key distribution from computational assumptions**
Tony Metger, Yfke Dulek, Andrea Coladangelo, and Rotem Arnon-Friedman.
ArXiv preprint.

Acknowledgments

A lot of credit for the existence of this dissertation goes to my primary advisor, Christian Schaffner. He taught me almost everything I know about quantum cryptography, and gave tons of feedback on my ideas and writings. I especially appreciate the way that Chris thoughtfully selected responsibilities and opportunities for me, urging me to take on challenges I wasn't sure I was ready for, but also advising me to take a step back and breathe when necessary. He would happily help me practice a talk or proofread a text, but would also sometimes say that he did not need to because he was certain that it was already good. I enjoyed exploring the PhD-supervision process together, and am sure that many more students will happily follow in my footsteps.

As my second advisor, Harry Buhrman complemented Chris's supervision by taking on the role of an experienced mentor. He gave me guidance through conversations on ethics and career paths, created opportunities for me to speak and to connect with people in his network, and always cheered me on. Harry also poured a lot of his energy into creating QuSoft, which has proven to be an extremely fruitful environment to learn and do science.

One aspect of doing research that I have come to appreciate immensely is collaboration with others: brainstorming together at the whiteboard, bouncing ideas around, asking (and, sometimes, answering) a thousand stupid and smart questions, sharing happiness if a construction seems to work, and sharing disappointment if it breaks down again. I am grateful to all of the great people I have had the chance to co-author papers with: Alex Grilo, for getting as excited as me when something seems to *almost* work, and for being a walking Complexity Zoo; Andrea Coladangelo, for being one of the very few people that can genuinely laugh in amazement when an idea is disproven, and just be happy to understand things better because of it; Christian Majenz, for his endless patience in explaining the more mathematical aspects of quantum information, and for being able to do so even if there was no piece of paper available because we were biking or on the train (also, for his tips on how to make the *best* pumpkin pie); Florian Speelman, for the many whiteboard pens we emptied together, for the times that our brains seemed in sync when trying to tackle the problem at hand, and for his eerily accurate intuition of when something is (or is not) going to work; Gorjan Alagic, for asking me in Darmstadt whether we could build quantum obfuscation, for continuing down the rabbit hole of projects that arose from that one question with me, and for being one of my biggest supporters; Rotem Arnon-Friedman, for setting a superb example by doing everything she does with 130% commitment, and for inviting me to Berkeley and welcoming me into her home; Stacey Jeffery, for her ability to break up seemingly insurmountable research questions into manageable bites, and for sharing her experiences and advice on being a WIQD; Tony Metger, for his attitude of getting things done instead of only talking about it, and for finding compromises between differing opinions; and Zvika Brakerski, for his fierce enthusiasm about any and all technical puzzles, and for taking the time to talk me through his thought processes even if those were going a thousand miles per second.

The result of all those collaborations is this book. I would like to thank the members of my thesis committee for taking the time to read and assess its content. I look forward to discussing any and all of its details with Gorjan Alagic, Serge Fehr, Stacey Jeffery, Cees de Laat, Sonja Smets, and Ronald de Wolf.

My time as a PhD student has also benefited greatly from the people that I was surrounded by on a daily basis. A comfortable work environment was ensured by Jenny and Tanja at the ILLC, and Susanne and Doutzen at CWI. I enjoyed being surrounded by the many excellent staff, students, and postdocs that walk the halls of CWI every day: Alex, Álvaro, Arjan, András, Bas, Chris, Chris, Chris, Farrokh, Florian, Freek, Harold, Harry, Ido, Jan, Jana, Jeroen, Jonas, Jop, Joran, Joris, Koen, Lars, Māris, Michael, Peter, Ronald, Sander, Simon, Srinivasan, Stacey, Subhasree, Teresa, Tom, and Yinan deserve thanks for the seminars, lunches, coffee-machine chats, yoga sessions, working sessions at the QuSoft corner, open stage nights, board games, train rides, pepernoten, and of course the many, many foosball matches. I look forward to seeing many of them around in the coming years, although it will take some time until Covid-19 has passed and we can see each other in person again. In the meantime, let us keep up the online seminars, QuTeas, and group meetings from the safety of our own homes. (Chris, I am afraid we will just have to assume that I would have passed the can-you-beat-your-supervisor-in-foosball test.) Several new people have joined in the past year, and I look forward to seeing them in person as well: Dmitry, Jordi, Marten, Mehrdad, Philip, René, Sebastian, and Yanlin, welcome to QuSoft!

I am grateful for my awesome friends, who kept me happy (and somewhat sane) whenever I was not at the office. They urged me to relax on evenings, weekends, and holidays, but let me sneak in an important work email or two when I thought they weren't looking. I want to thank them for all the talks, walks, climbing sessions, skiing trips (and aftermovies), swimming marathons, TanCKI barbecues, board games, virtual games, road trips, dinners, potlucks, celebrations, hugs, creative poems, cups of tea, Advents of Code, needlessly philosophical discussions, and cat pictures. I am curious to see where we will be in another four years, and hope to keep all of you around. Geerten deserves a special mention for his endless supply of quantum jokes, which are simultaneously funny and horrible (pun intended).

My family has always been around to cheer me on. I especially want to thank my grandmother, who has the magical ability to steer the universe in a positive direction

just by twiddling her thumbs; my late grandfather, a shining example of what any scientist should strive to become; and of course my big brother, who pulled out our toy blackboard when he was six, and decided to teach me the alphabet (it all starts somewhere). But most importantly, my parents, who have always encouraged me to pursue what feels right, and have been supportive, loving and proud every single day for the past 28 years, no matter where my path has led me. The drive from Leiden to Utrecht is never too far for my father, and a phone call “just to chat” never comes at an inconvenient moment for my mother. They have made sure I know there is always a safe place to return to.

And finally, my deepest gratitude is owed to my partner Camiel. A single paragraph simply cannot do justice to the amount of encouragement, devotion, and all-around happiness he has given me. Even though he has no way of knowing whether the formulae I write down are groundbreaking or utter nonsense, Camiel has had this fierce conviction that I am going to become the best scientist the world has ever seen. Whenever I face a difficult problem or scary deadline, he convinces me that I have the power to make it through. And when the time comes to say “I told you so”, he always says it kindly and proudly (of me, and a little of himself because he is right yet again). Camiel, I am so very fortunate to have had you by my side for these past nine years, and for all the adventures that are yet to come.

Yfke Dulek
Utrecht, November 2020.

1 Introduction

Chapter contents

1.1 Secret messages

Humans like secrets. We like making secrets, keeping secrets, and finding out other people's secrets. From a recipe for a pottery glaze to military plans to personal letters [Kah96], we have attempted to shield our writings from prying eyes all throughout history.

Initially, hiding a message involved nothing more than writing it down in a slightly complicated way. The sender would, for example, replace certain symbols with less common ones, or with ones with a different meaning but a similar sound. Or perhaps she would write some words backwards while leaving out vowels, or replace them with obscure jargon.

These message-hiding techniques may have been sufficient in a time where written communication was anyway reserved for a small fraction of the population. However, in principle, anyone with the ability to read and a knack for puzzles would be able to figure out the message eventually. The techniques lacked a certain ingredient that is now considered essential to cryptography: a *key*, a secret piece of information that is needed to decode the ciphertext back into the original message.

In Roman times, the first known cipher requiring such a key emerged. In the Caesar cipher, the key specifies the number of places to transpose each letter in the message. If it is 2, for example, every `A` in the message becomes a `C` in the ciphertext, every `B` becomes a `D`, et cetera. Despite the secret key, a lot of information about the underlying message is maintained by this method of encryption: by counting the frequency of each letter, observing repeating patterns, or even just trying all 26 different possible keys, it would still be fairly easy to recover the message.

In the centuries that followed, increasingly complicated ciphers were invented. The idea was that more intricate combinations of transpositions, substitutions, and permutations would lead to ciphers that were harder to break. The receiver of a message would need to remember more secret steps in order to successfully decode the message, and for an eavesdropper, guessing all of these steps would become increasingly difficult. Effectively, concatenating several types of ciphers increased the size of the secret key.

In 1883, Kerckhoffs observed that a proper secret key is crucial to a successful cryptographic protocol. He postulated that the security of a protocol should not rely on obscurity of the inner workings of the protocol itself, but instead only on the fact that an eavesdropper does not know the secret key [Ker83]. Kerckhoffs's principle has been essential to shaping cryptography as we know it today.

Shortly after the Second World War, which featured many newly designed and quickly broken cryptographic protocols, Shannon formalized the notion of *information content* of a message [Sha48]. He viewed information as a quantifiable resource that can be expressed in bits. To properly hide a message, one has to ensure that there is no information about the message present in the ciphertext. At the same time, a receiver that knows the secret key should of course be able to decipher all the information about the message. Shannon showed that the only way to simultaneously achieve these properties is to use a secret key of the same length as the message [Sha49].

The canonical example of an information-theoretically secure encryption scheme is the one-time pad. Encrypting an n-bit message $x \in \{0,1\}^n$ with the one-time pad requires an n-bit secret key $k \in \{0,1\}^n$. The ciphertext $c \in \{0,1\}^n$ is defined by simply taking the bitwise XOR between the message and the key: $c_i := x_i \oplus k_i$ for every $1 \leqslant i \leqslant n$. If the key is chosen uniformly at random, one can prove that the message x and ciphertext c share no information: the one-time pad provides perfect *information-theoretic* security.

1.1.1 Public-key cryptography

Information-theoretically secure encryption using a secret key that is as long as the message is not very practical in most cases: in order to securely send an n-bit message, one would first have to communicate an n-bit key to the receiver. Public-key cryptography [DH76; RSA78] reduces the amount of secret key by slightly relaxing what it means to keep a message secret: instead of requiring that the ciphertext contains no information about the message, it is only required that the information is *hard to find*. This idea is usually formalized by stating that if one is able to recover the message from the ciphertext, then one is also able to solve a specified mathematical problem. Under the assumption that this problem is hard to solve, it should be practically impossible to break the encryption. This type of security is called *computational security* (or security under computational assumptions), and is almost always used in practice.

Public-key cryptography relates to an important question in theoretical computer science: which problems are hard? The cryptographic systems in use today are based on problems which are *believed* not to be solvable by a computer in a reasonable (i.e., polynomial in the input length) amount of time, such as integer factorization [RSA78] or discrete logarithms of elliptic curves [DH76]. The possibility always remains that a new, efficient algorithm for one of those mathematical problems is discovered, rendering the cryptographic protocols based on them insecure.

One may wonder why cryptosystems cannot just be designed based on problems that are *known* to be hard (e.g., an EXP-complete problem), instead of working with problems that are on the edge of being solvable by an efficient algorithm. This is unlikely to be a successful endeavor, because the underlying mathematical problem needs to provide enough structure to allow for efficient honest usage: encryption should be efficient in practice, and so should decryption for someone holding the secret key. Thus, cryptosystem design requires us to walk a fine line between efficiency for the user and computational intractability for the attacker.

Enter the quantum computer. This relatively new model of computation shakes the foundations of our knowledge about which problems are hard and which are not. Although not yet implemented for large inputs in practice, Shor showed that a quantum computer has the ability to factor integers efficiently [Sho94], thereby breaking cryptosystems that are based on the assumption that integer factorization is hard.

To stay ahead of potential quantum adversaries, and secure our communications going forward, we need to identify mathematical problems that are hard enough to resist quantum attacks, but structured enough to support meaningful cryptosystems. The field of *post-quantum cryptography* is concerned with finding such problems, analyzing them, and designing cryptosystems around them. Currently, there are several promising candidates almost ready to be implemented, but also still very actively under review by the scientific community [AAA+20]. Post-quantum secure encryption, although not the main topic of this dissertation, will feature as a prominent building block of many of the protocols throughout this work.

1.1.2 Quantum cryptography

Although they form a threat for the security of many cryptographic protocols, the hope is that quantum computers will become useful for a wide array of computational applications. Quantum computing can potentially enhance search and optimization algorithms [Gro96; BS17], speed up the development of chemical products such as drugs [LD12] and fertilizers [RWS+17], and maybe even aid artificial intelligence [DB18].

Once commercial-scale quantum computers become available that can provide these applications, they will not exist in isolation. In contrast to the classical computer, the quantum computer will be born into the age of the Internet, big data, and commercial computing. We need to be careful to securely store and communicate quantum data right from the start.

In the design of protocols to encrypt quantum messages, one has to essentially start from scratch. This is because the structure of quantum data differs fundamentally from that of classical data: while a classical message is usually modeled as a string of bits (0 or 1), a quantum message consists of qubits, a much richer object that can be in one of infinitely many different states.

In the search for quantum-cryptographic protocols that are simultaneously efficient and secure, the possibility of information-theoretic security came into view again [BB84]. Quantum mechanics can potentially circumvent the limitations inherent to classical cryptographic tasks. Indeed, by making clever use of quantum entanglement and the uncertainty principle, it is possible to design information-theoretically secure quantum protocols for tasks such as key distribution [BB84; Eke91], secret sharing [CGL99; Got00], and fingerprinting [BCWW01].

A notable example of an information-theoretically secure quantum protocol is the *quantum one-time pad* [AMTW00], which securely encrypts quantum messages. Similarly to the classical one-time pad, the secret key for the quantum one-time pad is fairly large: to encrypt a message consisting of n qubits, one needs a secret key of length $2n$. The crucial fact that makes the quantum one-time pad a useful tool, however, is the fact that the secret key can be *classical.* Thus, the quantum one-time pad reduces the task of privately sending n qubits to communicating a classical key of $2n$ bits. Since qubits are considered to be a much more precious resource than classical bits, at least for the foreseeable future, this reduction can be quite powerful. The quantum one-time pad therefore plays a prominent role in many protocols that require some form of privacy, like the protocols in this dissertation.

1.2 Delegated and distributed quantum computation

Around the turn of the millennium, it became apparent that even though quantum mechanics provided possibilities that were out of reach with just classical techniques, some cryptographic primitives could still not be realized information-theoretically. Among them are fundamental primitives such as bit commitment [May97; LC97; LC98], coin tossing [LC97], and oblivious transfer [Lo97]. Some more complex tasks, which could in theory be achieved via many different routes, turned out to be impossible without computational assumptions as well: two examples that will be relevant to this dissertation are two-party quantum computation [BCS12] and quantum fully homomorphic encryption [YPF14].

Precisely those more complex tasks, where a computation is distributed amongst multiple parties, or delegated to a computationally more powerful party, are desirable in a quantum setting. A potential near-future scenario is one where a few universities, institutes and companies around the globe will have the capability to build and operate a full-scale quantum machine. It is not very likely that such machines will be available to and affordable for the general public, or even to the average company or government, in the short term. For that, quantum computers simply require too much expertise and precise equipment to operate. To facilitate the use of quantum computers by other parties, they could be hooked up onto a (classical or quantum) internet. Third parties, interested in using the quantum capabilities to perform computations, could "log on" to the quantum computer via some secure cloud service, and delegate their computation to this quantum server.

Even for classical computations, the above scenario is much more complex than merely sending a message over a network. When sending a message, one needs to protect against external parties that may eavesdrop (or even actively try to sabotage the protocol), but the sender and receiver can be trusted. In this new setting, however, we also need security against *internal* adversaries: the participants of a distributed or delegated computation do not necessarily trust each other, but they still have to work together to achieve a certain task. For instance, a client buying the services of a cloud-computing provider needs to send that provider certain information to perform the computation, but at the same time, may want to keep their inputs, outputs, and perhaps even computations secret from the provider. For such scenarios, a protocol needs to provide all parties involved with enough information to appropriately perform their assigned tasks, but not so much information that they are able to break the security of the protocol.

To complicate matters further, quantum computers and networks are still in active development, causing a moving target for quantum cryptographers. Should a protocol be optimized for the amount of communication, memory, or computation steps? As long as we do not know which will be the most expensive commodity in quantum computing, it is hard to tell which trade-offs are worth it and which are not. The only safe assumption that we can make is that classical computation and communication will remain easier than their quantum variants for quite some time.

In summary, protocol design for delegated and distributed quantum computation is challenging for a variety of reasons. Firstly, we are still exploring which cryptographic primitives are even achievable: those that are impossible information-theoretically may still be possible under computational assumptions. Secondly, the desired applications are relatively young in classical cryptography as well, and therefore not yet well understood. Often, the quantum primitive is a generalization of the classical one, and there is no reason to assume that it will be simpler to achieve. Finally, it is not always clear by what metric to judge the efficiency of quantum protocols.

1.2.1 Related work

Despite the uncertainties mentioned above, the field of delegated and distributed quantum computation has steadily been moving forward over the last fifteen years.

Childs initiated the study of securely delegating quantum computations [Chi05]. He observed that the quantum one-time pad has a certain structure: some physical operations on the ciphertext always correspond to certain logical operations on the plaintext message, regardless of the value of the classical secret key. By analyzing the commutation relations between the one-time pad and certain quantum operations, Childs designed a protocol which allowed a client to partially outsource a quantum computation to a more powerful server. This protocol is still highly interactive, and the client has to do a fair amount of computation (both classical and quantum) that depends on the secret key.

In subsequent developments, the situation for the client was improved in a variety of ways. Several protocols reduced the workload for the client by limiting it to only classical operations [RUV13; CGJV19] (sometimes requiring the presence of a second server that cannot communicate with the first one). Others reduced the amount of communication required between client and server [FHM18; Gri17], or provided additional perks for the client, such as privacy [BFK09; DFPR14] or verifiability [ABE10; Mor14; Bro15; HM15; FK17] of the executed quantum program.

Despite these developments, the boundary of what is possible to achieve information-theoretically has to be respected. At that point, it is time to turn to a solution that has proven to be very powerful classically: the computational assumption. Primitives that may be hard or impossible to achieve against unbounded adversaries, have been found to be achievable if the adversaries are computationally bounded.

Interestingly, in the quantum setting, computational assumptions are usually not woven directly into the structure of the quantum encryption. Instead, the security of computationally secure quantum-cryptographic primitives is based on a two-step mechanism. First, the security is reduced to the classical realm, such as a classical secret key or a classical computation. Then, the classical variant of that primitive is employed to ensure security. If the latter is secure only under computational assumptions, then so is the resulting quantum scheme.

As a prime example, consider the quantum one-time pad, which itself provides information-theoretically secure encryption of quantum states, using only a classical key k: let us call the encryption procedure $\mathsf{QOTP.Enc}_k$. To turn this private-key scheme into a public-key one, we can encrypt the classical key k under a public-key encryption scheme $\mathsf{Classical.Enc}$ (with independent public key pk and secret decryption key sk), and attach the encrypted secret key to the ciphertext. The encryption of a quantum state $|\psi\rangle$ then becomes:

$$\mathsf{QOTP.Enc}_k\left(|\psi\rangle\right) \otimes \left|\mathsf{Classical.Enc}_{pk}(k)\right\rangle . \tag{1.1}$$

Note that the computational assumption only comes in on the classical level: this type of hybrid between an information-theoretical quantum scheme and a computational classical scheme will appear several times throughout this work.

1.3 Contributions

The goal of this dissertation is to explore the possibilities and impossibilities of various cryptographic primitives for delegated and distributed quantum computation. We will mainly focus on the possibilities that computational assumptions bring beyond what can be achieved information-theoretically. Specifically, we will consider the following three quantum-cryptographic primitives:

Quantum homomorphic encryption (Chapters 5 and 6). A homomorphic encryption can be used to delegate a quantum computation noninteractively while preserving secrecy of the input to the computation. The requirement of noninteractivity forms the main hurdle in the design of a (quantum or classical) homomorphic-encryption scheme: the client cannot use his knowledge about the secret key to "guide" the server during the computation. Instead, the server must perform the entire computation without knowledge of the secret key.

We overcome this hurdle by providing the server with a state that depends on the secret key, thereby allowing the server to extract the client's directions herself. We take this one step further by enabling the server to construct a *proof* that the computation was performed correctly, so that the client can verify the homomorphic computation at decryption time.

Multi-party quantum computation (Chapter 4). In multi-party quantum computation, several players (each with their own quantum input) want to perform a joint computation without fully trusting each other. In particular, they do not want to reveal their inputs to each other, and they want to ensure that the computation is not sabotaged by another player. In this distributed-computation setting, the difficulty is that there is no single trusted party that can generate and hold the secret key: in principle, all players are potentially adversarial.

We solve this difficulty by observing that, in its core, it is again a *classical* problem: through classical multi-party computation, the players can agree on, keep track of, and choose actions based on a classical secret key. This way, we "lift" multi-party classical computation to multi-party quantum computation.

Quantum obfuscation (Chapter 7). In obfuscation, the roles are reversed compared to delegated computation: the client (or "user") has sufficient computational power to run a quantum computation, but is not allowed to learn the specification of the computation. The server (or "obfuscator") hands out an obfuscated version of the program, to which the user can choose his own input. In this setting, the obfuscator holds the secret information. The difficulty is that this obfuscator is not involved in the computation after encrypting the program: thus, the user needs to be able to not only perform the computation, but also *decrypt the result*, without access to the secret information or communication with the obfuscator.

The initial goal for this dissertation was to "lift" a protocol for classical obfuscation to a quantum protocol, using the verifiable quantum homomorphic encryption scheme we construct. Instead, we show that quantum obfuscation in its strongest sense is impossible to achieve, even under computational assumptions. It may still be possible to lift weaker variants of obfuscation from the classical to the quantum setting; we leave that as a future direction to explore.

A recurring theme in the three cryptographic primitives just discussed is *verification*: the client wants to verify that the server performed a homomorphic computation correctly, the players want to verify that no other player sabotaged the multi-party computation, and the obfuscator wants to build in a verification procedure that only reveals the plaintext output if the obfuscation was used in the intended way.

In this dissertation, we explore how this verification property can be achieved using tools from *message authentication*. Traditionally, a message authentication code is used to ensure that a message arrives at a receiver unaltered. In other words, the receiver can *verify* that after being encoded by the sender, the message has only undergone the identity operation. (Quantum) computing on authenticated data extends this idea to allow the receiver to verify that a specific nonidentity operation has been applied to the message instead. As such, the message authentication code necessitates the application of a specified operation by the untrusted server (or other untrusted party).

In reminiscence of the fine line we have to walk between efficiency for the user and intractability for the attacker when designing encryption schemes, quantum computing on authenticated data needs to find a balance between structure (allowing an honest server to perform the specified computation) and lack thereof (preventing a dishonest server from performing unauthorized computations). Certain more complex verification primitives require even stronger properties from the underlying message authentication code. We investigate properties such as integrity of the ciphertext and reusability of the key in Chapter 3.

1.3.1 Techniques

The general security framework that we will work in is the so-called *real-vs.-ideal* paradigm. In an ideal scenario, the adversary or eavesdropper is very limited in what it can do to break privacy or sabotage a computation: this scenario reflects the situation we want to achieve. The real scenario is the actual protocol, with no restrictions on the adversary. By showing that the real scenario is indistinguishable from the ideal one, we prove that the real protocol achieves the desired level of security.

We opt for the real-vs.-ideal paradigm because of the composability it provides: once a real protocol is proven indistinguishable from a certain ideal one, that ideal protocol can then be used as a subprotocol in other, more complex settings. At that point, we do not have to concern ourselves with the implementation details of the subprotocol, but instead can model it as an ideal interaction. This approach makes our results directly useful for future research.

To reinforce the composable and modular nature of our results, we will often turn to proofs by reduction. That is, many of our protocols are black-box upgrades from either classical or simpler quantum primitives. The advantage of this approach is twofold: (1) it simplifies the analysis by isolating the elements that are required for the upgrade, allowing us to focus exclusively on those elements in the proof; (2) improvements of the underlying simpler primitives directly benefit our constructions without having to augment the proofs. For example, if the underlying simpler primitive can be based on a simpler or more promising computational assumption, then so can our quantum primitives. This flexibility is especially useful since it is currently unclear which computational assumptions will turn out to be realistic.

A downside of the modular approach is that the security and efficiency bounds are likely not tight. At every composition step, general bounds are combined (multiplied or added), whereas more specific bounds could potentially be achieved if the specific protocol is analyzed in its entirety. However, we stress that the aim of this dissertation is not to construct practically implementable cryptographic primitives. Rather, our contribution is to show that certain quantum-cryptographic primitives are achievable (or unachievable) on a theoretical level. For that goal, general asymptotic bounds suffice.

2 Preliminaries

Chapter contents

2.1 Introduction

No twenty-first-century dissertation exists in isolation, and this one does not either. It is set within existing frameworks, and some of the notation and techniques have been developed in previous work. This chapter discusses several of these preliminary concepts. It by no means substitutes a thorough introduction into the fields of cryptography and quantum computing: there already exist excellent books and lecture notes that achieve that goal [KL14; NC00; Wat11; Wol19]. Instead, the goal of this chapter is to establish notation and conventions, and to highlight a number of concepts that will be central to this work.

2.2 Theoretical computer science

2.2.1 Sets and strings

Sets are denoted by Roman capital letters. The number of elements in a set S is $|S|$, and can possibly be infinite. The intersection $S \cap T$ contains only those elements that are in both S and T. The difference $S \setminus T$ contains the elements that are in S, but not in T. The union $S \cup T$ contains the elements that are in S or T, or both. If we want to stress that S and T are known to be disjoint, then we write $S \sqcup T$ for their union. We will use the notation $S \subsetneq T$ for strict subsets, and $S \subseteq T$ if it is possible that $S = T$.

Several specific sets will appear often in this work.

The blackboard-bold letters $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ denote the set of natural numbers, group of integers, field of real numbers, and field of complex numbers, respectively. The notation $[n]$ refers to the set $\{1, 2, \ldots, n\}$.

The set $\{0,1\}^n$ contains all possible n-bit strings. Given such a string x, we write x_i to denote the ith bit of that string. In situations where that notation may be confusing (for example, when talking of sequences $(x_i)_{i \in [m]}$ of strings), we use the notation or $x_i[j]$ for the jth bit of the string $x_i \in \{0,1\}^n$. The *Hamming weight*, i.e., the number of ones in x, is denoted $|x|$. The set $\{0,1\}^*$ contains all bit strings of unspecified length, including the empty string, that is, $\{0,1\}^* = \bigcup_{i=0}^{\infty} \{0,1\}^i$. For two strings $x, y \in \{0,1\}^n$, we write $x \oplus y$ for the bitwise XOR, and $x \cdot y$ for their scalar product $\sum_{i=1}^{n} x_i \cdot y_i \pmod 2$. By convention, 0 denotes the all-zero string 0^n whenever n is clear from context.

Another set of special interest is the Galois field of two elements, $\mathbb{F}_2$, with additive identity 0, and multiplicative identity 1. Addition corresponds to a logical XOR (denoted $\oplus$), with identity element 0. Multiplication corresponds to a logical AND (denoted $\cdot$), with identity element 1.

For any field F, $GL(n,F)$ is the general linear group of degree n over F. It consists of all $n \times n$ invertible matrices with elements from F as their entries, with the group operation being matrix multiplication.

In Chapter 3, we will use the following general upper bound on the number of sets in a family of sets, given a restriction on the sizes of their intersections.

Lemma 2.2.1 (Ray-Chaudhuri–Wilson inequality [RW75])**.** *Let $\mathcal{F}$ be a family of subsets of some set of n elements, with each subset containing exactly k elements. Let $L = \{l_1, \dots, l_s\}$ be a collection of allowed intersection sizes (with $l_i \leqslant k$ for all i). If $\mathcal{F}$ is L-intersecting, i.e., $\left|\mathcal{F}_i \cap \mathcal{F}_j\right| \in L$ for all $i \neq j$, then*

$$|\mathcal{F}| \leqslant \binom{n}{s}.$$

In the above lemma, $\binom{n}{s}$ denotes the binomial coefficient.

2.2.2 Algorithms and protocols

An algorithm is a set of instructions for a computer that, given an input, follows a set of basic steps to produce some output. Most of this work will be set in the circuit model, where the algorithm is represented by a circuit C consisting of some set of elementary gates, that represent very primitive functionalities, like addition or multiplication. Unless otherwise specified, we assume that the circuits can be generated uniformly for all input sizes. That is, there exists an efficient Turing machine or circuit that, given n as input, produces a circuit description C_n that handles inputs of size n.

We consider an algorithm to be *efficient* if it runs in time polynomial in the size of its input. In the circuit model, this means that the size of the circuit is polynomial in the input size. We usually do not specify the exact polynomial, but simply write $\mathrm{poly}(n)$ to denote the running time.

An algorithm A can be probabilistic, which is modeled by a deterministic algorithm A_{det} such that the output distribution $\Pr_r\left[A_{\mathsf{det}}(x, r)\right]$ equals the distribution of $A(x)$, for all x. Most of the classical algorithms featured in this dissertation are probabilistic polynomial-time (PPT) algorithms.

When combining circuits, we write $C_2 \circ C_1$ for the composition of two circuits C_1 and C_2: first, the circuit C_1 is run on the input, and its output is fed into C_2, similarly to function composition.

A *protocol* is a collection of algorithms in a specific setting. For example, an encryption protocol consists of separate algorithms for key generation, encryption, and decryption (see Section 2.3.1). Protocols can involve multiple parties and even be interactive. Depending on the setting, it may be possible to run one protocol right after another. In that case, we write $\Pi_2 \diamond \Pi_1$ for the composition of the two protocols Π_1 and Π_2. For an algorithm $\mathcal{A}$ interacting with a protocol Π, potentially with several rounds of inputs and outputs, we write $\mathcal{A} \leftrightarrows \Pi$ for the final output of $\mathcal{A}$ after the entire interaction.

2.2.3 Probabilities

If D is a distribution, we write $x \leftarrow D$ to signify that x is sampled according to D, or $x \leftarrow_D S$ if we want to be explicit about the universe S on which the distribution is defined. For a finite set S, we write $x \leftarrow_R S$ to signify that x is sampled uniformly at random from the set S.

For a distribution D on some finite universe S, the expected value of a real function f is written as

$$\mathop{\mathbb{E}}_{x \leftarrow D} f(x) := \sum_{x \in S} \Pr[x \leftarrow D] \cdot f(x). \tag{2.1}$$

If the distribution D is clear from the context, we may simply write $\mathbb{E}_x$ instead of $\mathbb{E}_{x \leftarrow D}$.

Probabilities are implicitly taken over the randomness of any probabilistic algorithms inside the expression. For example, $\Pr_x [A(x) = 1]$ is actually a shorthand for $\Pr_{x,r} [A_{\mathsf{det}}(x, r) = 1]$, where A_{det} is the deterministic algorithm corresponding to A.

The statistical distance between two distributions D and D' over the same finite universe S is defined as

$$\frac{1}{2} \sum_{x \in S} \left| \Pr[x \leftarrow D] - \Pr[x \leftarrow D'] \right|. \tag{2.2}$$

We write $D \approx_\varepsilon D'$ to signify that the statistical distance between D and D' is upper bounded by ε.

In certain cases, we want to express that even though two distributions are not statistically close, they are indistinguishable to any efficient algorithm. Two distribution ensembles $\{D_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{D'_\lambda\}_{\lambda \in \mathbb{N}}$ (where the λ functions as a security parameter; see Section 2.3.1) are computationally indistinguishable, written $D_\lambda \overset{c}{\approx} D'_\lambda$, if no polynomial-time algorithm can distinguish between a sample from one distribution or the other. That is, for all PPT A,

$$\left| \Pr_{x \leftarrow D_\lambda} [A(x) = 1] - \Pr_{y \leftarrow D'_\lambda} [A(y) = 1] \right| \leqslant \mathrm{negl}(\lambda). \tag{2.3}$$

We sometimes write $x \overset{c}{\approx} y$ if it is clear from which distributions x and y are sampled. If not even an efficient *quantum* algorithm can distinguish them, the distributions are quantum computationally indistinguishable. We use the same notation for the quantum setting.

2.3 Cryptography

2.3.1 Encryption: correctness and security

A classical encryption protocol (or scheme) consists of three efficient algorithms: a key generation algorithm KeyGen, an encryption algorithm Enc, and a decryption algorithm Dec. If the protocol is symmetric (or private-key), then the same (secret) key k is used by both Enc and Dec: we write $\mathsf{Enc}_k(x)$ for the encryption of a message x, and similarly, $\mathsf{Dec}_k(y)$ for the decryption of a ciphertext y. An *asymmetric* (or public-key) protocol generates two different keys: a public key pk for the encryption, and a secret key sk for decryption. In a public-key scheme, anyone can encrypt a message, but only the party holding a secret key should be able to decrypt.

Any encryption protocol should be *correct*: for all x, $\mathsf{Dec}_k(\mathsf{Enc}_k(x)) = x$. A protocol that is correct for all keys is perfectly correct. Sometimes, we may accept a small correctness error, i.e., it may be acceptable that $\mathsf{Dec}_k(\mathsf{Enc}_k(x)) = x$ only with high probability over k.

For security, we take an asymptotic approach. There is a *security parameter* $\lambda \in \mathbb{N}$, which is usually given as a unary argument to KeyGen. Intuitively, we require that the honest executions of KeyGen, Enc, and Dec run in time polynomial in λ, but that *breaking* the encryption (in a sense that we will specify below) only succeeds with probability negligible in λ.

Definition 2.3.1. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for all constants $c \in \mathbb{N}$ there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, $|f(n)| < n^{-c}$.

A negligible function is (eventually) smaller than any inverse polynomial. We let $\mathrm{negl}(\lambda)$ denote an unspecified negligible function. The definition of a negligible function can be extended by stating that $f(n)$ is negligible in another function $g(n)$ if for all $c \in \mathbb{N}$, there exists an n_0 such that for all $n \geqslant n_0$, $|f(n)| < (g(n))^{-c}$.

A negligible probability of successfully breaking a primitive is deemed a secure margin, because an adversary that can try polynomially many times (e.g., by interacting polynomially many times with the primitive) still cannot break the primitive with a noticeable probability.

2.3.2 Game-based security

One possible way to characterize security is by defining a *security game*, which is played between an honest challenger and a potentially dishonest adversary. A protocol for a certain cryptographic primitive is deemed secure if no adversary can win the game with more than a negligible advantage.

The game-based security framework is very well suited for protocols that are defined in terms of one or more already-existing cryptographic primitives. One can often proceed by reduction, proving a statement of the following form: *"If there exists* *an adversary that can win the security game with a more-than-negligible advantage, then there exists another adversary that can win the security game for the other cryptographic primitive with a (related) more-than-negligible advantage. Since we know the latter to be impossible, security follows."* In other words, the game-based security framework connects the security of a protocol to that of lower-level protocols.

One of the most basic security games for encryption is the indistinguishability game, which is played as follows[1], a (for a private-key scheme):

1. The challenger runs $k \leftarrow \mathsf{KeyGen}$.
2. The adversary, on input 1^λ, outputs a message x. (This message is called the *challenge plaintext.*)
3. The challenger samples a random bit $b \leftarrow_R \{0,1\}$. If $b = 1$, he computes the ciphertext $c \leftarrow \mathsf{Enc}_k(x)$. Otherwise, he computes $c \leftarrow \mathsf{Enc}_k(0^{|x|})$.
4. The adversary receives c (the *challenge ciphertext*), and outputs a bit b'.

If the adversary manages to output $b' = b$, we say that she "wins" the game.

Definition 2.3.2 (IND)**.** A private-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is IND-secure if no adversary can properly distinguish between an encryption of x and of $0^{|x|}$. That is, for any probabilistic adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A} \text{ wins the indistinguishability game}\right] \leqslant \frac{1}{2} + \mathrm{negl}(\lambda).$$

The probability is taken over the randomness inside the game (from the key generation, encryption, and selection of the bit b), and the randomness used by $\mathcal{A}$.

Aside from IND, there are several common game-based security notions for encryption protocols. They are increasingly demanding: IND-CPA ("indistinguishability under chosen-plaintext attacks"), allows the adversary to request a polynomial number of plaintexts to be encrypted before having to choose the challenge plaintext. IND-CCA1 ("indistinguishability under chosen-ciphertext attacks"), additionally allows the adversary to request decryptions of arbitrary ciphertexts before selecting the challenge plaintext. Finally, in IND-CCA2, the adversary may even request decryptions *after* receiving the challenge ciphertext, but is of course not allowed to request decryption of the challenge ciphertext c itself.

For public-key encryption, analogous notions of IND, IND-CPA, IND-CCA1, and IND-CCA2 can be defined. In this setting, the adversary receives the public key as input before having to select the challenge plaintext. Using this public key, she can

[1]Conventionally, the indistinguishability game is stated in a slightly different way [KL14]: the adversary chooses two messages, x_0, and x_1, and the challenger encrypts x_b. This formulation of the indistinguishability game is equivalent to the one presented here.

simulate her own encryption oracle, rendering IND and IND-CPA equivalent. In the public-key setting, we often only require security to hold if the adversary $\mathcal{A}$ is a PPT algorithm (see Section 2.3.4).

2.3.3 Semantic security (real-vs.-ideal)

Another way to characterize security is within the so-called real-vs.-ideal framework. In this framework, an *ideal functionality* is defined in which the information flow is very limited. Anything that an adversary can learn or do in an interaction with the actual protocol (the "real world") should be simulatable in an interaction with this ideal functionality (the "ideal world"). No environment should be able to tell the difference between the two worlds, not even if it has control over the plaintext.

As an example, consider the private-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The environment, through a message-generation algorithm $\mathcal{M}$, generates a message x, along with any extra side information s. In the real world, an adversary $\mathcal{A}$ receives $(\mathsf{Enc}_k(x), s)$. In the ideal world, the simulator $\mathcal{S}$ receives only s. Both the adversary and the simulator produce an output: the environment tries to distinguish between those two outputs using a distinguisher algorithm $\mathcal{D}$. The semantic security notion SEM states that the probability that the environment succeeds is very small:

Definition 2.3.3 (SEM)**.** A private-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is SEM-secure if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$, such that for all environments $(\mathcal{M}, \mathcal{D})$,

$$\left|\Pr\left[\mathcal{D}\left(\mathcal{A}\left(\mathsf{Enc}_k(x), s\right)\right) = 1\right] - \Pr\left[\mathcal{D}\left(\mathcal{S}(s, 1^\lambda)\right) = 1\right]\right| \leqslant \mathrm{negl}(\lambda),$$

where $(x, s) \leftarrow \mathcal{M}(1^\lambda)$. The probability is taken over KeyGen, Enc, $\mathcal{A}$, $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{D}$.

In other words, the statistical distance between $\mathcal{A}(\mathsf{Enc}_k(x), s)$ and $\mathcal{S}(s, 1^\lambda)$ should be negligible. Again, in a public-key setting, we often only require Definition 2.3.3 to hold for efficient adversaries $\mathcal{A}$: in that case, the simulator and environment should also be efficient, and the message generator $\mathcal{M}$ receives the public key as input.

Definition 2.3.3 is equivalent to Definition 2.3.2 [KL14]. However, semantic security notions have two major advantages over game-based security notions. First, it is often clearer that the semantic definition really captures the desired level of security: in SEM, for example, it is directly clear that whatever you are able to learn when given the ciphertext $\mathsf{Enc}_k(x)$, you can also learn without it. Second, semantic security notions are (sequentially) *composable*: because they take into account the environment (including any side information) so explicitly, they can more easily be integrated into higher-level cryptographic primitives. If one proves that a certain protocol satisfies some semantic notion of security, one can then safely use the ideal functionality as a building block for more complex protocols, without worrying about the exact implementation of it.

Game-based security and semantic security each have their pros and cons. In this dissertation, we will use notions from both security frameworks, proving them to be equivalent where applicable.

2.3.4 Computational assumptions

The notions of security discussed in Sections 2.3.2 and 2.3.3 are unconditional, or information-theoretic: no adversary, no matter how powerful or patient, can succeed in breaking the encryption. Because of its formulation in terms of a security parameter and a statistical distance, these notions are also called *statistical*.

In practice, however, we often rely on computational assumptions. As discussed in the introduction, computational assumptions allow for more efficient encryption schemes, and can in some cases circumvent impossibility results.

In a computational security notion, the adversary (and, if applicable, the simulator and environment) is a polynomial-time algorithm. Security statements are phrased in terms of some computational problem P, parametrized by the security parameter λ: "*Under the assumption that* $P(\lambda)$ *is hard, no efficient adversary can break the protocol with probability more than* $\text{negl}(\lambda)$."

Traditionally, the security of cryptographic schemes is based on problems such as integer factorization (where λ determines the size of the integer to be factorized), or computing the discrete logarithm over certain groups (where λ determines the order of the group). These problems, although still thought to be hard for classical computers, can be solved efficiently on a quantum computer [Sho94]. Therefore, we need to shift our attention to cryptographic protocols based on different computational assumptions.

In this dissertation, we will work with the *learning-with-errors* (LWE) problem as a computational assumption. There, the problem is to find a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, given polynomially many samples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, with $\mathbf{a} \leftarrow_R \mathbb{Z}_q^n$, and $e \leftarrow_\chi \mathbb{Z}_q$. The size (or dimension) n, modulus q, and distribution χ all depend on the security parameter λ. A typical choice of parameters is $n = \text{poly}(\lambda)$, $q = \text{poly}(\lambda)$, and χ the normal distribution with mean 0 and standard deviation $1/\text{poly}(\lambda)$ [Reg10]. For a weaker assumption, one can choose q to be exponential in λ, but this may jeopardize the efficiency of the cryptographic scheme.

2.4 Quantum computing

2.4.1 States

Quantum information is carried by *qubits* which can be in a certain state. Pure states are represented by length-1 vectors $|\psi\rangle$ (or $|\varphi\rangle, \ldots$) in a complex Hilbert space $\mathcal{H}$. Mixed states, represented by density operators ρ (or $\sigma, \mu, \nu, \ldots$) on that Hilbert space, are more general and can represent mixtures of pure quantum states. The state space $\mathfrak{D}(\mathcal{H})$ contains all possible density operators, i.e., the set of Hermitian, positive semi-definite matrices with trace 1. As a special case, a pure quantum state $|\psi\rangle$ can be represented as a density operator $|\psi\rangle\langle\psi|$ (where $\langle\psi|$ is the adjoint of $|\psi\rangle$).

There are several well-known quantum states that have their own notation. The computational-basis states are represented as $|0\rangle$ or $|1\rangle$, or, for a higher-dimensional system, $|s\rangle$ for some string $s \in \{0,1\}^d$. Note that $|s\rangle$ is an abbreviation for $|s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_d\rangle$. In the Hadamard basis, the basis states are written $|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. We reserve the special state $|\perp\rangle$ as the "reject" state, which is orthogonal to all other states[2]. The EPR (Einstein-Podolsky-Rosen) pair $|\Phi^+\rangle$ is the entangled two-qubit state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The other three Bell pairs are written as $|\Phi^-\rangle := \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$, $|\Psi^+\rangle := \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$, and $|\Psi^-\rangle := \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. Finally, we reserve the symbol τ for the completely mixed state $\mathbb{I}/d$.

2.4.2 Gates

Quantum states are computed upon, in general, by completely positive trace preserving (CPTP) maps (or "quantum channels"), generally written as capital Greek letters $\Phi, \Psi, \Lambda, \ldots$, or as a calligraphic letter referring to their role in a protocol, such as $\mathcal{A}$ (for adversary) or $\mathcal{S}$ (for simulator). We sometimes abuse notation and give a classical input x to a quantum algorithm, writing $\Phi(x)$: in that case, the algorithm Φ is actually given the computational-basis state $|x\rangle$ as input.

A subcategory of CPTP maps is formed by the (reversible) unitary operations, denoted with the letters $U, V, W, \ldots$. The complex conjugate of U is written $U^\dagger$. For an n-bit string $s = s_1 s_2 \cdots s_n$, define $U^s := U^{s_1} \otimes U^{s_2} \otimes \cdots \otimes U^{s_n}$ (where $U^1 = U$ and $U^0 = \mathbb{I}$).

For a projector Π onto a subspace of the Hilbert space, we write $\overline{\Pi}$ for its complement $\mathbb{I} - \Pi$.

We work in the quantum circuit model, with circuits C composed of elementary unitary gates, preparation of auxiliary inputs in the state $|0\rangle$, and computational-basis measurements. We consider those measurement gates to be destructive, i.e., to destroy the post-measurement state immediately, and only a classical wire to remain. Since subsequent gates in the circuit can still classically control on those measured wires, this point of view is as general as keeping the post-measurement states around after a computational-basis measurement.

For a set of quantum gates $\mathcal{G}$, the $\mathcal{G}$-depth of a quantum circuit is defined as the minimal number of layers such that in every layer, gates from $\mathcal{G}$ do not act on the same qubit.

For a quantum circuit C, we write Φ_C for the channel induced by C. We consider the channel Φ_C to run in quantum polynomial time (QPT) if the circuit C has size

[2]This can straightforwardly be modeled using a single extra qubit to expand the dimension of the state.

polynomial in the number of input wires (and can be efficiently generated from that number, see Section 2.2.2). Conversely, every efficient quantum map can be represented by a polynomial-size circuit that initializes all noninput wires to $|0\rangle$ before applying unitary gates and computational-basis measurements.

To enable a clear discussion of the different parts of a circuit, we group several wires together into quantum registers (denoted $M, R, S, T, \dots$). We write $|R|$ for the dimension of the Hilbert space underlying a register R: that is, a register R containing n qubits has $|R| = 2^n$. The registers in which a certain quantum state exists, or on which some channel acts, are written as gray superscripts whenever they may be unclear otherwise. For example, a unitary U that acts on register A, applied to a state ρ in the registers AB, is written as $U^A \rho^{AB} U^\dagger$, where the registers on which $U^\dagger$ acts can be determined by finding the matching U and reading the gray subscripts. Note that we do not explicitly write the operation $\mathbb{I}^B$ with which U is in tensor product. The gray superscripts are purely informational, and do not signify any mathematical operation. If we want to denote, for example, a partial trace of the state ρ^{AB}, we use the conventional notation ρ_A.

The standard set of elementary gates that we will use throughout this work is known as the "Clifford+T" set. It consists of three "levels": the Pauli group, the Clifford group (which is a superset of the Pauli group), and the T gate.

Pauli group. The single-qubit Pauli group $\mathscr{P}_1$ is, up to global phase, generated by the bit flip and phase flip operations,

$$\mathsf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathsf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

The X and Z operators anti-commute ($\mathsf{XZ} = -\mathsf{ZX}$). Up to a global phase, all four single-qubit Pauli operators are of the form $\mathsf{X}^a\mathsf{Z}^b$ with $a, b \in \{0,1\}$. We generally ignore the global phase of a quantum state, as it is not observable by measurement.

The m-qubit Pauli group $\mathscr{P}_m$ consists of m-fold tensor products of single-qubit Paulis. Using the notation introduced earlier, we can write $\mathsf{X}^a\mathsf{Z}^b =: P_{a,b}$ with $a, b \in \{0,1\}^m$ to denote an element of $\mathscr{P}_m$ (again, up to a global phase). For notational brevity, we may refer to Pauli operators as P_ℓ, where ℓ can either be understood to be a number between 0 and 2^{2m-1}, or a $2m$-bit string. By convention, P_0 refers to the identity operation (i.e., $a = b = 0^m$).

The *weight* of a Pauli is the number of locations i in which it is nonidentity, i.e., where $a_i \vee b_i = 1$. We can also talk about the X-weight (defined as the Hamming weight $|a|$) or the Z-weight (defined as $|b|$) separately.

The identity gate is an element of the Pauli group, and is written as I (for a single qubit) or $\mathsf{I}^{\otimes m}$ (for m qubits). It equals the identity matrix $\mathbb{I}$ of dimension 2^m.

Clifford group. The m-qubit Clifford group $\mathscr{C}_m$ consists of all unitaries U that commute with the Pauli group as a whole: that is, the Clifford group is the normalizer of the Pauli group. Since all m-qubit Pauli operators are of the form $\mathsf{X}^a\mathsf{Z}^b$, this means that U is a Clifford operator if for any $a,b \in \{0,1\}^m$ there exist $a',b' \in \{0,1\}^m$ such that (up to a global phase):

$$U\mathsf{X}^a\mathsf{Z}^b = \mathsf{X}^{a'}\mathsf{Z}^{b'}U.$$

All Pauli operators are easily verified to be elements of the Clifford group (in that case, $a',b' = a,b$). The entire Clifford group is generated [Got98] by

$$\mathsf{P} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad \mathsf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \text{and } \mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.4}$$

In the literature, the P gate is also referred to as the phase gate or the S gate, or sometimes the K gate. It relates to the phase-flip gate as $\mathsf{P}^2 = \mathsf{Z}$.

Three consecutive CNOT gates (with alternating control and target wires) result in a SWAP gate, which swaps the contents of two wires. Generalizing, for any n-element permutation $\pi \in S_n$, the unitary that permutes n wires according to π is also a Clifford operation:

$$\pi\left(|\psi_1\rangle \cdots |\psi_n\rangle\right) = |\psi_{\pi(1)}\rangle \cdots |\psi_{\pi(n)}\rangle. \tag{2.5}$$

We have the following commutation relations between elements of the Pauli group and the Clifford group:

$$\mathsf{PX} = i\mathsf{XZP}, \tag{2.6}$$
$$\mathsf{PZ} = \mathsf{ZP}, \tag{2.7}$$
$$\mathsf{HX} = \mathsf{ZH}, \tag{2.8}$$
$$\mathsf{HZ} = \mathsf{XH}, \tag{2.9}$$
$$\mathsf{CNOT}(\mathsf{X}\otimes\mathsf{I}) = (\mathsf{X}\otimes\mathsf{X})\mathsf{CNOT}, \tag{2.10}$$
$$\mathsf{CNOT}(\mathsf{I}\otimes\mathsf{X}) = (\mathsf{I}\otimes\mathsf{X})\mathsf{CNOT}, \tag{2.11}$$
$$\mathsf{CNOT}(\mathsf{Z}\otimes\mathsf{I}) = (\mathsf{Z}\otimes\mathsf{I})\mathsf{CNOT}, \tag{2.12}$$
$$\mathsf{CNOT}(\mathsf{I}\otimes\mathsf{Z}) = (\mathsf{Z}\otimes\mathsf{Z})\mathsf{CNOT}. \tag{2.13}$$

In situations where global phase is irrelevant, we may leave it out in Equation (2.6).

Whenever a protocol mandates handing an element from the Clifford group to an agent, we mean that a (classical) description of the group element is given, e.g. as a normal-form circuit. That means that if a protocol defines the Clifford element as a product of several other group elements, those individual group elements are not necessarily revealed to the receiving party.

Because of its definition in terms of the Pauli group, the Clifford group has an intricate relation with it. Given any two nonidentity Paulis, there are a number of Cliffords that map the first Pauli to the second Pauli by conjugation. This number is always the same:

Lemma 2.4.1 (Pauli partitioning by Cliffords [ABEM17]). *For every $P, Q \in \mathscr{P}_n \backslash \{\mathsf{I}^{\otimes n}\}$,*

$$\left|\left\{C \in \mathscr{C}_n \mid C^\dagger P C = Q\right\}\right| = \frac{|\mathscr{C}_n|}{|\mathscr{P}_n| - 1}.$$

As a corollary, every nonidentity Pauli is mapped to a *random* nonidentity Pauli when conjugated by a random Clifford.

Corollary 2.4.2 (Clifford randomization). *For every $P \in \mathscr{P}_n \backslash \{\mathsf{I}^{\otimes n}\}$,*

$$\frac{1}{|\mathscr{C}_n|} \sum_{C \in \mathscr{C}_n} C^\dagger P C = \frac{1}{|\mathscr{P}_n| - 1} \sum_{Q \in \mathscr{P}_n \backslash \{\mathsf{I}^{\otimes n}\}} Q.$$

Proof. By definition of the Clifford group, $C^\dagger P C = Q$ for some $Q \in \mathscr{P}_n$. The proof is then a straightforward derivation using Lemma 2.4.1:

$$\frac{1}{|\mathscr{C}_n|} \sum_{C \in \mathscr{C}_n} C^\dagger P C = \frac{1}{|\mathscr{C}_n|} \sum_{Q \in \mathscr{P}_n} \left|\left\{C \in \mathscr{C}_n \mid C^\dagger P C = Q\right\}\right| Q \tag{2.14}$$

$$= \frac{1}{|\mathscr{C}_n|} \sum_{Q \in \mathscr{P}_n} \frac{|\mathscr{C}_n|}{|\mathscr{P}_n| - 1} Q \tag{2.15}$$

$$= \frac{1}{|\mathscr{P}_n| - 1} \sum_{Q \in \mathscr{P}_n \backslash \{\mathsf{I}^{\otimes n}\}} Q. \tag{2.16}$$

□

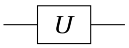
T gate. The Clifford group itself does not suffice to perform arbitrary quantum computations: in fact, Clifford computations on classical inputs are classically simulatable [Got99]. However, by adding any non-Clifford gate, any quantum circuit can be efficiently computed with only a small error. We choose this non-Clifford gate to be the T gate (also known as the $\pi/8$ gate),

$$\mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{2.17}$$

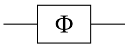
Note that the T gate, because it is non-Clifford, does not commute with the Pauli group. More specifically, we have $\mathsf{T}\mathsf{X}^a\mathsf{Z}^b = e^{-\pi i/4}\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}$. Again, in situations where the global phase is irrelevant, we may leave it out.

Pictorial representations of circuits are often instructive. In figures, the circuit input is on the left, and the output on the right. Gates are applied from left to right. Single wires represent quantum data, whereas double wires represent classical information. For more high-level circuits, a single wire may represent one or more registers, which may actually consist of many wires.

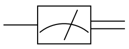
Below, we list several common elements in circuit figures. Any additional elements will be introduced in the caption of the figure in question.

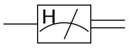


unitary or gate (2.18)

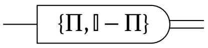


map or algorithm (2.19)

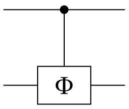

measurement (computational basis) (2.20)



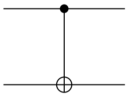
measurement (Hadamard basis) (2.21)

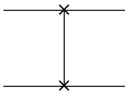


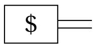
projective measurement (2.22)



controlled gate/unitary/map/algorithm (2.23)


CNOT (controlled-X) (2.24)


SWAP (2.25)



fresh, uniform randomness (2.26)

2.4.3 Norms and distances

Let $\| \cdot \|_2$ be the vector 2-norm, i.e., $\| \, |\psi\rangle \, \|_2 := \sqrt{\langle\psi|\psi\rangle} = \sqrt{\sum_i \alpha_i \alpha_i^*}$ for a vector $|\psi\rangle = \sum_i \alpha_i \, |i\rangle$.

For matrices, we will encounter the Schatten p-norm for $p = 1, 2, \infty$: for finite p, this norm (on a matrix A) is defined as

$$\| A \|_p := \left(\mathrm{Tr} \sqrt{\left(A^\dagger A\right)^p} \right)^{1/p} . \tag{2.27}$$

The case $p = 1$ is known as the trace norm, and we write $\| A \|_{\mathrm{tr}} := \| A \|_1 = \mathrm{Tr}\sqrt{A^\dagger A}$. It satisfies the triangle inequality, ($\| A + B \|_{\mathrm{tr}} \leqslant \| A \|_{\mathrm{tr}} + \| B \|_{\mathrm{tr}}$), Hölder's inequality (specifically, $\| AB \|_{\mathrm{tr}} \leqslant \| A \|_2 + \| B \|_2$), submultiplicativity ($\| AB \|_{\mathrm{tr}} \leqslant \| A \|_{\mathrm{tr}} \cdot \| B \|_{\mathrm{tr}}$), and multiplicativity with respect to the Kronecker product ($\| A \otimes B \|_{\mathrm{tr}} = \| A \|_{\mathrm{tr}} \cdot \| B \|_{\mathrm{tr}}$).

For $p = \infty$, the Schatten norm $\| A \|_\infty := \lim_{p \to \infty} \| A \|_p$ is the operator norm induced by the vector 2-norm. That is, for $A : \mathcal{H}_1 \to \mathcal{H}_2$,

$$\| A \|_\infty = \sup_{\substack{|\psi\rangle \in \mathcal{H}_1 \\ \| |\psi\rangle \|_2 = 1}} \| A |\psi\rangle \|_2 . \tag{2.28}$$

It satisfies multiplicativity w.r.t. the Kronecker product: $\| A \otimes B \|_\infty = \| A \|_\infty \cdot \| B \|_\infty$.

In Chapter 4, we will use the following variant of Hölder's inequality, modified for a partial trace:

Lemma 2.4.3 ([Maj20]). *For a matrix A with domain X, and matrix B with domain XY,*

$$\left\| \mathrm{Tr}_X \left[A^X B^{XY} \right] \right\|_{\mathrm{tr}} \leq \left\| A^X \right\|_\infty \cdot \left\| B^{XY} \right\|_{\mathrm{tr}} . \tag{2.29}$$

Proof. Using an alternative characterization [Wat11, Equation (1.173)] of the trace norm, $\| E \|_{\mathrm{tr}} = \sup_F |\mathrm{Tr}[FE]| \cdot \| F \|_\infty^{-1}$, we derive

$$\begin{aligned}
\left\| \left| \mathrm{Tr}_X \left[A^X B^{XY} \right] \right| \right\|_{\mathrm{tr}} &= \sup_C \frac{\mathrm{Tr}\left[C^Y \left(\mathrm{Tr}_X \left[A^X B^{XY} \right] \right) \right]}{\left\| C^Y \right\|_\infty} && (2.30) \\
&= \sup_C \frac{\left| \mathrm{Tr}\left[\left(A^X \otimes C^Y \right) B^{XY} \right] \right|}{\left\| C^Y \right\|_\infty} && (2.31) \\
&= \left\| A^X \right\|_\infty \cdot \sup_C \frac{\left| \mathrm{Tr}\left[\left(A^X \otimes C^Y \right) B^{XY} \right] \right|}{\left\| A^X \right\|_\infty \cdot \left\| C^Y \right\|_\infty} && (2.32) \\
&= \left\| A^X \right\|_\infty \cdot \sup_C \frac{\left| \mathrm{Tr}\left[\left(A^X \otimes C^Y \right) B^{XY} \right] \right|}{\left\| A^X \otimes C^Y \right\|_\infty} && (2.33) \\
&\leq \left\| A^X \right\|_\infty \cdot \sup_D \frac{\left| \mathrm{Tr}\left[D^{XY} B^{XY} \right] \right|}{\left\| D^{XY} \right\|_\infty} && (2.34) \\
&= \left\| A^X \right\|_\infty \cdot \left\| B^{XY} \right\|_{\mathrm{tr}} . && (2.35)
\end{aligned}$$

□

The trace norm has many applications across quantum information theory. In this dissertation, we will mainly use it to express the trace distance $\frac{1}{2} \left\| \rho - \sigma \right\|_{\mathrm{tr}}$, a measure for how different two mixed states ρ and σ are. Note that if ρ and σ are classical mixtures of computational-basis states, their trace distance amounts to the statistical distance between the classical distributions. We write $\rho \approx_\varepsilon \sigma$ whenever the trace distance is upper bounded by ε.

The diamond norm of a channel Φ acting on the input state space $\mathfrak{D}(\mathcal{H})$ is written as $\|\Phi\|_\diamond := \sup_{\rho \in \mathfrak{D}(\mathcal{H}^{\otimes 2})} \left\| (\mathbb{I} \otimes \Phi)(\rho) \right\|_{\mathrm{tr}}$. Here, the identity serves to take into account any additional registers with which the input state of Φ may be entangled. If we want to talk about the distance between two channels Φ and Ψ, we use the normalized quantity $\frac{1}{2}\|\Phi - \Psi\|_\diamond$, which we refer to as the *diamond-norm distance*. Similarly to the trace distance, we may write $\Phi \approx_\varepsilon \Psi$ to express an ε upper bound to the diamond-norm distance.

2.4.4 Magic-state computation

In some contexts, applying non-Clifford gates is not straightforward for different reasons: common quantum error-correcting codes do not allow transversal[3] implementation of non-Clifford gates, and non-Clifford gates do not commute with the Pauli group which is used in many quantum encryption schemes.

In order to concentrate the hardness of non-Clifford gates in an offline pre-processing phase, we can use "magic-state computation" [BK05]: if we are able to prepare so-called *magic states* of the form $|\mathsf{T}\rangle := \mathsf{T}\,|+\rangle$, then we do not need the ability to apply a T gate directly. Using a single copy of this state as a resource, we are able to implement a T gate using the circuit in Figure 2.1. The circuit only requires (classically controlled) Clifford gates and measurements.

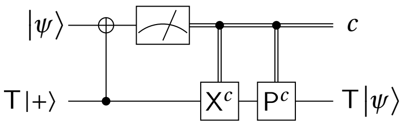


Figure 2.1: Using a magic state $|\mathsf{T}\rangle = \mathsf{T}\,|+\rangle$ to implement a T gate.

Magic-state computation is also possible for other gates than the T gate. By using a different magic state as a resource, and applying a different classically-controlled correction unitary, we can for example apply a phase gate P, or identity. Magic-state computation for other gates is discussed in more detail in Section 3.4.2.

2.4.5 Oracles

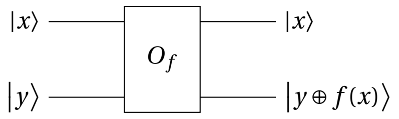
A (classical or quantum) algorithm can have access to an *oracle*: a black-box object that accepts inputs and computes a certain function on those inputs. The algorithm with oracle access can see the input/output behavior of the function, but not its inner workings. An oracle is often a convenient way to model some ideal functionality, which is guaranteed not to leak any unwanted information.

If an algorithm A has oracle access to some classical function f (or quantum map Φ), we write A^f (or A^Φ). If A has access to multiple oracles with separate input/output interfaces, we write, e.g., $A^{f,g}$.

[3]In quantum error correction, n physical qubits represent a single logical qubit. A gate U is transversal for the code if applying $U^{\otimes n}$ to the physical qubits has the same effect as applying U to the logical state.

The algorithm A has query depth d if its queries can be partitioned into d (finite) sets, where each set only depends on the outcomes of the previous queries. In this work, we will usually assume that A poses a single query at a time.

A quantum algorithm, when given a *classical* oracle, is always assumed to have superposition access to it. For a classical function $f : \{0,1\}^n \to \{0,1\}^m$, the superposition oracle acts as the following quantum map O_f:



(2.36)

The "one-way to hiding" lemma [Unr15] provides an upper bound on the probability that a quantum adversary can tell the difference between two different classical oracles f and g. Classically, this probability can be bounded by looking at the content of the queries: what is the probability that an adversary queries an input from the set on which f and g differ? Quantumly, the situation is more delicate: since the adversary can query in superposition, it can easily submit a query with nonzero weight on the differing set. The one-way to hiding lemma, stated below in its formulation by Ambainis, Hamburg, and Unruh [AHU19], claims that the probability can be bounded by measuring a random query in the computational basis, and observing whether the measurement result lies in the differing set.

Lemma 2.4.4 (One-way to hiding [AHU19, Theorem 3])**.** *Let $f, g : \{0,1\}^n \to \{0,1\}^m$ and $z \in \{0,1\}^*$ be sampled from an arbitrary joint distribution. Let $S := \{x \in \{0,1\}^n \mid f(x) \neq g(x)\}$. Let A be a quantum algorithm with oracle query depth d. Define B^f to be the following quantum algorithm: on input z, it picks $i \leftarrow_R \{1,2,\ldots,d\}$, and runs A^f until (just before) the ith query. It then measures the query input register in the computational basis, and outputs the measurement result. Then we have*

$$\left|\Pr\left[A^f(z)=1\right] - \Pr\left[A^g(z)=1\right]\right| \leq 2d\sqrt{\Pr\left[B^f(z) \in S\right]}.$$

By symmetry, Lemma 2.4.4 also holds if B is given oracle access to g instead of f.

2.4.6 Twirling

One of the techniques we use in this work is the twirl over a group $\mathcal{G}$ of unitary operators, which maps a state (or channel) to its '$\mathcal{G}$-averaged' version. Specifically, the twirl of a state ρ is defined as

$$\mathcal{T}_{\mathcal{G}}(\rho) := \frac{1}{|\mathcal{G}|}\sum_{U \in \mathcal{G}} U\rho U^\dagger, \tag{2.37}$$

and the twirl of a channel Λ is defined as

$$\mathcal{T}_{\mathcal{G}}(\Lambda(\cdot)) := \frac{1}{|\mathcal{G}|}\sum_{U\in\mathcal{G}} U^\dagger(\Lambda(U(\cdot)U^\dagger))U. \tag{2.38}$$

We sometimes abuse notation for nonunitary groups: for example, in Chapter 4, we use $\mathcal{T}_{GL(2n,\mathbb{F}_2)}(\cdot)$ to denote a twirl over the unitary group $\{U_g \mid g \in GL(2n,\mathbb{F}_2)\}$, where U_g is defined as the unitary that applies g in-place, i.e., $U_g\,|t\rangle = |g(t)\rangle$ for all $t \in \{0,1\}^{2n}$.

The Pauli twirl, i.e., the twirl of a channel over the Pauli group, is of special interest to us. The reason is that any unitary U can be written as a (weighted) sum of Pauli operators. More generally, if U acts on an n-qubit register A and some other register B, we can rewrite it as:

$$U^{AB} = \sum_{P\in\mathscr{P}_n} P^A \otimes U_P^B. \tag{2.39}$$

Here, the U_P are not normalized: the weighting of the Pauli P is absorbed into it.

Given the above decomposition of a unitary into a superposition of its Pauli components, we can see that the Pauli twirl has the effect of transforming that superposition of Pauli attacks into a *classical mixture* of Pauli attacks. This transformation greatly simplifies analysis in many cryptographic settings:

Lemma 2.4.5 (Pauli twirl of a channel [ABEM17, Lemma 5.1])**.** *For all unitaries $U^{AB} = \sum_{P\in\mathscr{P}_n} P^A \otimes U_P^B$,*

$$\mathcal{T}^A_{\mathscr{P}_n}(U(\cdot)U^\dagger) = \sum_{P\in\mathscr{P}_n} (P\otimes U_P)(\cdot)(P\otimes U_P)^\dagger,$$

where the twirl is applied on the 2^n-dimensional register A.

At the heart of the proof of Lemma 2.4.5 lies the following fact, which is also sometimes known as the Pauli twirl lemma:

Lemma 2.4.6 (Pauli twirl lemma [DCEL09])**.** *Let ρ be an arbitrary n-qubit state. Then for any $P, P' \in \mathscr{P}_n$, it holds that*

$$\frac{1}{2^{2n}}\sum_{Q\in\mathscr{P}_n} Q^\dagger P Q \rho Q^\dagger P'^\dagger Q = \begin{cases} P\rho P^\dagger & \text{if } P = P' \\ 0 & \text{otherwise} \end{cases}$$

Lemma 2.4.6 tells us that certain cross terms cancel out when twirling over the Pauli group. This cancellation effect is also captured by the following equality [Por17, Equation (2)], which effectively restates Lemma 2.4.6 in a different language. We will appeal to the formulation in Lemma 2.4.7 regularly in Chapter 3.

Lemma 2.4.7. *For all pairs $\ell = (a,b)$ with $a, b \in \{0,1\}^n$, we have*

$$\sum_{x,z\in\{0,1\}^n} (-1)^{\langle (x,z),\ell\rangle_{\mathsf{Sp}}} = \begin{cases} 2^{2n} & \text{if } \ell = (0^n, 0^n) \\ 0 & \text{otherwise.} \end{cases}$$

Here, the symplectic product $((x,z),(a,b))_{\mathsf{Sp}}$ *is defined as* $(x\cdot b)\oplus(z\cdot a)$ *(with* $\cdot$ *representing the scalar product modulo 2).*

For the Clifford group, a statement similar to Lemma 2.4.5 holds. The cross terms (with $P \neq P'$) also cancel, but the Clifford elements remain in the final expression, because they cannot be trivially commuted out. Using Corollary 2.4.2, it is possible to simplify the expression below further if necessary.

Lemma 2.4.8 (Clifford twirl of a channel [ABEM17]). *For all* $U^{AB} = \sum_{P\in\mathscr{P}_n} P^A \otimes U_P^B$ *where* U *is unitary,*

$$\mathcal{T}^A_{\mathscr{C}_n}(U(\cdot)U^\dagger) = \sum_{P\in\mathscr{P}_n} \mathop{\mathbb{E}}_{C\in\mathscr{C}_n} (C^\dagger PC\otimes U_P)(\cdot)(C^\dagger PC\otimes U_P)^\dagger,$$

where the twirl is applied on the 2^n*-dimensional register* A*.*

2.4.7 Quantum encryption

The majority of the cryptographic framework presented in Section 2.3 generalizes to the quantum case, except that we are now interested in encrypting *quantum plaintexts* into *quantum ciphertexts*. The triple $(\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ thus consists of quantum algorithms, although the keys are usually assumed to be classical, so that they can easily be reused. The security notions IND, IND-CPA, IND-CCA1, and SEM generalize to the quantum setting [BJ15; ABF+16] by allowing the adversary, simulator, message generator and distinguisher to be quantum algorithms, and the challenge plaintext to be quantum as well. Importantly, the adversary may generate a challenge plaintext that is entangled with her own registers. A quantum version of IND-CCA2 is more subtle [AGM18; CEV20], because one needs to define what it means to query the challenge ciphertext to the decryption oracle. We often write q-IND (q-IND-CPA, et cetera) to stress the fact that the security notion is quantum.

A widely used (information-theoretically secure) quantum encryption protocol is the quantum one-time pad (QOTP) [AMTW00]. For a single-qubit state ρ, it works as follows. The key generation selects two random bits $a,b \leftarrow_R \{0,1\}$. Encryption and decryption are both the application of the Pauli $\mathsf{X}^a\mathsf{Z}^b$. From the point of view of an adversary that does not know the secret key (a,b), the resulting state contains no information about the original message ρ, since

$$\sum_{a,b\in\{0,1\}} \left(\frac{1}{4}\mathsf{X}^a\mathsf{Z}^b\rho\left(\mathsf{X}^a\mathsf{Z}^b\right)^\dagger\right) = \tau, \qquad (2.40)$$

where τ is the fully mixed state.

Generalizing this idea to n qubits, we see that encryption is actually a twirl of the message ρ over the n-qubit Pauli group $\mathscr{P}_n$. This twirl always results in the fully mixed state:

Lemma 2.4.9 (Pauli twirl of a state)**.** *For all n-qubit quantum states ρ,*

$$\mathcal{T}_{\mathscr{P}_n}(\rho) = \tau.$$

Proof. Write $\rho = \sum_{i,j\in\{0,1\}^n} \alpha_{ij} |i\rangle\langle j|$. For every i, j we have

$$\mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) = \mathop{\mathbb{E}}_{x,z\in\{0,1\}^n} \mathsf{X}^x\mathsf{Z}^z |i\rangle\langle j| \mathsf{Z}^z\mathsf{X}^x \tag{2.41}$$

$$= \mathop{\mathbb{E}}_{x,z\in\{0,1\}^n} (-1)^{z(i\oplus j)} |i\oplus x\rangle\langle j\oplus x|. \tag{2.42}$$

Note that $\mathbb{E}_{z\in\{0,1\}^n}(-1)^{z(i\oplus j)} = 0$ whenever $i \neq j$ (i.e., $i \oplus j \neq 0$), and that the term evaluates to 1 whenever $i = j$. So

$$\mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) = \begin{cases} \mathbb{E}_x |i\oplus x\rangle\langle i\oplus x| = \tau & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2.43}$$

To conclude the proof, sum all terms of ρ and use $\sum_i \alpha_{ii} = \mathrm{Tr}(\rho) = 1$ to get

$$\mathcal{T}_{\mathscr{P}_n}(\rho) = \sum_{i,j} \alpha_{ij} \mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) = \sum_i \alpha_{ii}\tau = \tau. \tag{2.44}$$

□

Since the Pauli group is a subgroup of the Clifford group, Lemma 2.4.9 also holds when twirling over the n-qubit Clifford group.

The quantum one-time pad naturally appears after a quantum teleportation: if a state $|\psi\rangle$ is teleported through an EPR pair $|\Phi^+\rangle$, the resulting state is $\mathsf{X}^a\mathsf{Z}^b|\psi\rangle$, where (a, b) is the outcome of the teleportation Bell measurement. As long as the sender of the teleportation does not communicate (a, b) to the receiver, that receiver does not know anything about $|\psi\rangle$. In some interactive settings, we adopt the view of teleporting a state through an EPR pair to perform the QOTP encryption, rather than applying the unitary $\mathsf{X}^a\mathsf{Z}^b$ directly.

3
Quantum message authentication

Chapter contents

3.1 Introduction

A central topic in cryptography is authentication: how can we make sure that a message remains unaltered when we send it over an insecure channel? How do we protect a file from being corrupted when it is stored someplace where adversarial parties can potentially access it? And, especially relevant in the current era of cloud computing, can we let an untrusted third party compute on such authenticated data?

Following extensive research on authentication of classical data, which started with the seminal work by Wegman and Carter [WC81], several schemes have been proposed for authenticating quantum states [BCG+02; BCG+06; ABEM17; BGS13; GYZ17]. Barnum et al. [BCG+02] built schemes for the authentication of quantum data based on quantum error-correcting codes that are *purity testing*, meaning that any bit or phase flip on the message is detected with high probability.

Working in the abstract-cryptography framework, Portmann [Por17] showed that if the underlying code satisfies a stronger requirement called *strong purity testing*, the resulting authentication scheme provides “total authentication” [GYZ17], where no information about the key is leaked if the client accepts the authentication. This stronger guarantee allows for complete key recycling in the accept case, and even for partial key recycling in the reject case.

In a different direction, Alagic, Gagliardoni, and Majenz [AGM18] define a notion of quantum ciphertext authentication (QCA), where also the integrity of the ciphertext is guaranteed, and not just that of the plaintext. Ciphertext authentication is incomparable with total authentication: neither one implies the other (see Section 3.2.1).

A notable quantum message-authentication code is the trap code [BGS13], which surrounds the data with dummy qubits that function as *traps*, revealing any unauthorized attempts to alter the plaintext. The trap code is very well suited for quantum computing on authenticated data (see Section 3.4): it was originally designed for its use in quantum one-time programs [BGS13], but has found further applications in zero-knowledge proofs for QMA [BJSW16], and in quantum homomorphic encryption with verification [ADSS17].

The extraordinary structure of the trap code, which allows quantum computing on its ciphertexts, is simultaneously its weakness: an adversary can learn information about the secret key by altering the ciphertext in a single location, and observing whether or not the result is accepted by the client. Thus, to ensure security after de-authentication, the key needs to be refreshed before another quantum state is authenticated. More specifically, the error-correcting code that underlies the trap code is purity testing, but not strong purity testing. The need for a key refresh for every qubit inhibits the usefulness of the trap code as a code, because all qubits involved in a trap-code-authenticated computation need to be encoded under the same key to ensure correctness of the computation.

3.1.1 Contributions

In this chapter, we give an overview of existing quantum message-authentication codes, and the various definitions they may satisfy (Sections 3.2 and 3.3). We briefly describe how to compute on authenticated states without knowledge of the key (Section 3.4), as those techniques will be relevant for later chapters. We investigate the relation between (strong) purity testing and quantum ciphertext authentication (Sections 3.5 and 3.6), and give a variation on the trap code with stronger security guarantees (Section 3.7). We specify our new contributions in more detail below.

Quantum ciphertext authentication with key recycling (Section 3.2.1). We give a new definition for quantum authentication, QCA-R, that provides both ciphertext authentication and key recycling, and is strictly stronger than existing definitions.

Purity-testing codes result in QCA encryption (Section 3.5). We prove that Barnum et al.'s canonical construction of authentication schemes from purity-testing codes [BCG+02] produces schemes that are not only plaintext authenticating, but also ciphertext authenticating (QCA). The proof is a generalization of the plaintext-authentication proof [BW16], using a different (but still efficient) simulator. Note that our result immediately implies that the trap code is ciphertext authenticating.

Strong-purity-testing codes result in QCA-R encryption (Section 3.6). Purity-testing codes are generally not sufficient for constructing QCA-R schemes, but strong-purity-testing codes are: we prove that Barnum et al.'s canonical construction achieves QCA-R when a strong-purity-testing code is used as a resource. In case the authenticated message is accepted, the entire key can be reused. Otherwise, all but the quantum-one-time-pad key can be reused.

A strong-purity-testing version of the trap code (Sections 3.3.4 and 3.7). We give an explicit construction of a strong-purity-testing code that is inspired by the trap code. In this *strong trap code*, the underlying error-correcting code is not only applied to the data qubits, but also to the trap qubits. The result is a quantum authentication scheme which satisfies the strong notion of QCA-R, but still maintains the computational properties that make the original trap code such a useful scheme.

A constant-memory version of the trap code (Section 3.3.3). We present another variation on the trap code which has weaker security guarantees, but may be more useful in some practical settings. Its encoding and decoding procedures can be realized by a client with only a constant-sized quantum memory. On the downside, the code only provides plaintext authentication with an inverse-polynomial security error (versus an inverse-exponential error for the regular trap code).

Security under parallel encryption (Section 3.8). To illustrate the power of recycling key in the reject case, we consider a setting with a different type of key reuse: reusing (part of) a key immediately to authenticate a second qubit, even before the first qubit is verified. We show that, if multiple qubits are simultaneously authenticated using a scheme that is based on a strong-purity-testing code, then de-authenticating some of these qubits does not jeopardize the security of the others, even if their keys overlap. This property is especially important when using the computational capabilities of the strong trap code, since computing on authenticated qubits needs multiple qubits to use overlapping keys.

3.2 Definitions

3.2.1 Quantum message authentication

A quantum authentication code transforms a quantum state (the *logical* state or *plaintext*) into a larger quantum state (the *physical* state or *ciphertext*) in a way that depends on a secret key. An adversarial party that has access to the ciphertext, but does not know the secret key, cannot alter the logical state without being detected at decoding time.

Quantum authentication codes are syntactically the same as secret-key quantum encryption schemes (Section 2.4.7), and consist of three (efficient) algorithms: key generation KeyGen, encryption Enc_k, and decryption Dec_k. Throughout this chapter, we will assume for simplicity that KeyGen selects a key k uniformly at random from some set $\mathcal{K}$. However, our results still hold if the key is selected according to some other distribution.

A result by Alagic and Majenz [AM17, Lemma B.9] implies that we can characterize the encryption and decryption maps as being of the form

$$\mathsf{Enc}_k : \rho^{M} \mapsto U_k^{MT}\left(\rho \otimes \sigma_k^{T}\right)\left(U_k^{\dagger}\right)^{MT}, \tag{3.1}$$

$$\begin{aligned}\mathsf{Dec}_k : \rho^{MT} \mapsto \mathrm{Tr}_T &\left[\left(\Pi_k^{\mathsf{acc}}\right)^{T}\left(U_k^{\dagger}\rho U_k^{MT}\right)\left(\Pi_k^{\mathsf{acc}}\right)^{T}\right] + \\ &D_k^{MT}\left[\left(\Pi_k^{\mathsf{rej}}\right)^{T}\left(U_k^{\dagger}\rho U_k^{MT}\right)\left(\Pi_k^{\mathsf{rej}}\right)^{T}\right].\end{aligned} \tag{3.2}$$

Here, M is the message register, σ_k is some key-dependent *tag* state in register T, and U_k is a unitary acting on both. Π_k^{acc} and Π_k^{rej} are orthogonal projectors onto the support of σ_k and its complement, respectively. Finally, D_k is any channel: we will usually think of it as $D_k(\cdot) := \mathrm{Tr}_{MT}(\cdot) \otimes |\bot\rangle\langle\bot|^{M}$, i.e., it traces out the message and tag register entirely, and replaces the message with some fixed dummy state $|\bot\rangle$ that signifies a reject. Because of the above characterization, we will often talk about encryption schemes as a keyed collection $\{(U_k, \sigma_k)\}_{k\in\mathcal{K}}$ of unitaries and tag states.

Apart from hiding a message, one may require an encryption scheme to *authenticate* that message. There are several definitions of authentication of quantum data. All definitions involve some parameter ε, the security error. We usually require ε to be negligibly small in the size of the ciphertext.

The simplest definition is that of plaintext authentication, first defined by Barnum et al. [BCG+02], but stated here in a more general form that takes the possible presence of side information into account.

Definition 3.2.1 (Quantum plaintext authentication [DNS12])**.** A quantum encryption scheme $\{(U_k, \sigma_k)\}_{k \in \mathcal{K}}$ is *plaintext ε-authenticating* (or ε-DNS) if for all CP maps $\mathcal{A}$ (acting on the message register M, tag register T, and a side-information register R), there exist CP maps $\mathcal{S}_{\mathsf{acc}}$ and $\mathcal{S}_{\mathsf{rej}}$ such that $\mathcal{S} := \mathcal{S}_{\mathsf{acc}} + \mathcal{S}_{\mathsf{rej}}$ is trace-preserving, and

$$\frac{1}{2}\left\| \mathop{\mathbb{E}}_{k}\left[\mathsf{Dec}_k \circ \mathcal{A}^{MTR} \circ \mathsf{Enc}_k\right]^{MR} - \left(\mathbb{I}^M \otimes \mathcal{S}^R_{\mathsf{acc}} + |\bot\rangle\langle\bot|^M \left(\mathrm{Tr}_M \otimes \mathcal{S}^R_{\mathsf{rej}}\right)\right)\right\|_{\diamond} \leqslant \varepsilon,$$

where Enc_k and Dec_k are of the form of Equations (3.1) and (3.2).

The simulator in Definition 3.2.1 reflects the ideal functionality of an authentication scheme: in the accept case, the message remains untouched, whereas in the reject case, it is completely discarded and replaced with the fixed state $|\bot\rangle$. Any action on the side-information register R is allowed, as reflected by the fact that we pose no further constraints on $\mathcal{S}_{\mathsf{acc}}$ and $\mathcal{S}_{\mathsf{rej}}$.

Remark. All codes that will be discussed in Section 3.3 satisfy Definition 3.2.1.

A feature that is unique to quantum authentication is that any scheme satisfying Definition 3.2.1 automatically provides secrecy [BCG+02]. That is, any quantum authentication code is automatically also a quantum encryption scheme. The intuition behind this implication stems from the fact that being able to distinguish between two encoded basis states allows an adversary to disturb basis states in the conjugate basis. Turning the previous sentence around, if we want to ensure that the conjugate basis cannot be disturbed, we need to ensure that the original basis is properly encrypted. If we want to authenticate both the computational basis and the Hadamard basis, both need to be encrypted. In classical message authentication, there is only one basis to authenticate, and authenticating it does not necessarily imply encrypting that one basis.

Definition 3.2.1 specifies a form of so-called *one-time* authentication: the key k needs to be freshly random for every message that is sent. In reality, one might hope to use a single key k to send multiple messages. Under certain circumstances, reusing or recycling the key may be possible. As an alternative to Definition 3.2.1, Garg, Yuen and Zhandry define a notion they call “total authentication” [GYZ17]: in this definition, if verification accepts, the key can safely be recycled for another round of use. The definition below models this concept by revealing the key to the environment after verification, and requiring that it is indistinguishable from a completely fresh and uncorrelated key.

Definition 3.2.2 (Quantum plaintext authentication with key recycling [GYZ17])**.** A quantum encryption scheme $\{(U_k, \sigma_k)\}_{k \in \mathcal{K}}$ is *plaintext ε-authenticating with key recycling* (or ε-GYZ) if for all CP maps $\mathcal{A}$ (acting on the message register M, tag register T, and a side-information register R), there exist CP maps $\mathcal{S}_{\mathsf{acc}}$ and $\mathcal{S}_{\mathsf{rej}}$ such that $\mathcal{S} := \mathcal{S}_{\mathsf{acc}} + \mathcal{S}_{\mathsf{rej}}$ is trace preserving, and

$$\frac{1}{2}\left\| \mathop{\mathbb{E}}_{k}\left[\rho^{MR} \mapsto \mathrm{Tr}_T\left(\Pi_k^{\mathsf{acc}} U_k^\dagger \left(\mathcal{A}^{MTR}\left(U_k\left(\rho \otimes \sigma_k^T\right) U_k^\dagger\right)\right) U_k \Pi_k^{\mathsf{acc}}\right) \otimes |k\rangle\langle k|\right] - \left(\mathbb{1}^M \otimes \mathcal{S}_{\mathsf{acc}}^R \otimes \tau_{\mathcal{K}}\right)\right\|_\diamond \leqslant \varepsilon.$$

Note that Definition 3.2.2 only specifies what should happen in the accept case. It is possible to put constraints on the amount of key that is leaked in the reject case as well: we will see an example of how to do so in Definition 3.2.4. Either way, the above definition is strictly stronger than DNS authentication, meaning that all ε-GYZ authentication codes are also ε-DNS [AM17].

Quantum plaintext authentication with key recycling has been studied before. Oppenheim and Horodecki [OH05] showed partial key recycling for schemes based on purity testing codes, under a weaker notion of security. Hayden, Leung, and Mayers [HLM16] adapted Barnum et al.'s construction to use less key and show its authenticating properties in the universal-composability framework. Fehr and Salvail [FS17] developed a quantum authentication scheme for classical messages that achieves the same key-recycling rate as Portmann [Por17], but is not based on quantum error-correction and only requires the client to prepare and measure.

Strengthening Definition 3.2.1 in a different direction, Alagic, Gagliardoni, and Majenz introduced the notion of quantum ciphertext authentication [AGM18]. This notion does not limit the amount of key leaked, but requires that if authentication accepts, the entire *ciphertext* was completely untouched, rather than only the plaintext. This enhanced security is important for defining security notions such as a quantum analogue of IND-CCA2 security, where an adversary has access to a decryption oracle which it may query on any ciphertext other than the challenge ciphertext. Being able to slightly alter a ciphertext without altering the plaintext undermine the spirit of IND-CCA2. Classical message-authentication codes naturally have the ciphertext-authenticating property [AGM18].

Definition 3.2.3 (Quantum ciphertext authentication (QCA) [AGM18])**.** A quantum encryption scheme $\{(U_k, \sigma_k = \sum_r p_{k,r} |\varphi_{k,r}\rangle\langle\varphi_{k,r}|)\}_{k \in \mathcal{K}}$ is *ciphertext ε-authenticating* (or ε-QCA) if it is plaintext ε-authenticating as in Definition 3.2.1, and the accepting simulator $\mathcal{S}_{\mathsf{acc}}$ is of the form

$$\mathcal{S}_{\mathsf{acc}} : \rho^R \mapsto \mathop{\mathbb{E}}_{k',r}\left[\langle\varphi_{k',r}|^T \langle\Phi^+|^{M_1M_2}\, U^\dagger_{k'}\mathcal{A}^{M_1TR}\left(U^{M_1T}_{k'}\rho^{RM_1M_2T}_{k',r}U^\dagger_{k'}\right)U_{k'}\,|\varphi_{k',r}\rangle\,|\Phi^+\rangle\right],$$

where $\rho_{k',r} := \rho^R \otimes |\Phi^+\rangle\langle\Phi^+|^{M_1M_2} \otimes |\varphi_{k',r}\rangle\langle\varphi_{k',r}|^T$ is the input state before (simulated) encryption.

In QCA, the accepting simulator tests whether the message remains completely untouched by encrypting half of an EPR pair (in register M_1) as a "dummy message", under a key k' that it generates itself. It remembers the randomness r used in creating the tag state σ_k, so that it can accurately test whether the tag state was untouched.

In general, key recycling as in Definition 3.2.2 does not imply QCA [AM18]. To see this, take any scheme $\{(U_k,\sigma_k)\}_{k\in\mathcal{K}}$ that is plaintext authenticating with key recycling, and alter it by appending a qubit in the fully mixed state to σ_k (and extending U_k to act as identity on this qubit). This scheme still satisfies Definition 3.2.2. However, it cannot be ciphertext authenticating: attacks on this last qubit are not noticed in the real scenario, but the simulator $\mathcal{S}_{\mathsf{acc}}$ remembers the randomness with which this last qubit was chosen, and will test for it at decryption time.

Conversely, not all ciphertext-authenticating schemes have key recycling [AM18]. Take any scheme that is QCA, and alter it by adding one extra bit b of key, and setting $\sigma_{kb} := \sigma_k \otimes |b\rangle\langle b|$ and $U_{kb} := U_k \otimes \mathsf{I}$, effectively appending the (independent) bit of key at the end of the ciphertext. This scheme still satisfies Definition 3.2.3, but leaks at least one bit of key. For an overview of the relations between DNS, GYZ, and QCA, refer to Figure 3.1.

In addition to Definitions 3.2.1 to 3.2.3, we define a notion of quantum authentication that combines key recycling and ciphertext authentication. It is strictly stronger than the definitions mentioned before. In Section 3.6, we will show that Barnum et al.'s construction, when used with a strong-purity-testing code, results in an authentication scheme in this new, stronger sense.

Definition 3.2.4 (Quantum ciphertext authentication with key recycling (QCA-R)). A quantum encryption scheme $\{(U_k, \sigma_k = \sum_r p_{k,r}|\varphi_{k,r}\rangle\langle\varphi_{k,r}|)\}_{k\in\mathcal{K}}$ is *ciphertext ε-authenticating with key recycling* (or ε-QCA-R), with a key recycling function f, if for all CP maps $\mathcal{A}$ (acting on the message register M, tag register T, and a side-information register R), there exists a CP map $\mathcal{S}_{\mathsf{rej}}$ such that

$$\begin{aligned}\mathfrak{R} : \rho^{MR} \mapsto \mathbb{E}_k\Big[&\mathrm{Tr}_T\left(\Pi^{\mathsf{acc}}_k\left(U^\dagger_k\mathcal{A}^{MTR}\left(U^{MT}_k\left(\rho\otimes\sigma^T_k\right)U^\dagger_k\right)U_k\right)\Pi^{\mathsf{acc}}_k\right)\otimes|k\rangle\langle k|\\ &+ |\bot\rangle\langle\bot|^M \otimes \mathrm{Tr}_{MT}\left(\Pi^{\mathsf{rej}}_k\left(U^\dagger_k\mathcal{A}^{MTR}\left(U^{MT}_k\left(\rho\otimes\sigma^T_k\right)U^\dagger_k\right)U_k\right)\Pi^{\mathsf{rej}}_k\right)\otimes|f(k)\rangle\langle f(k)|\Big]\end{aligned}$$

is ε-close in diamond-norm distance to the ideal channel,

$$\mathfrak{I} : \rho^{MR} \mapsto \left(\mathbb{I}^M \otimes \mathcal{S}_{\mathsf{acc}}\right)\left(\rho^{MR}\right)\otimes\tau_{\mathcal{K}} \;+\; |\bot\rangle\langle\bot|^M \otimes \mathcal{S}_{\mathsf{rej}}(\rho^R)\otimes\mathbb{E}_k\left[|f(k)\rangle\langle f(k)|\right],$$

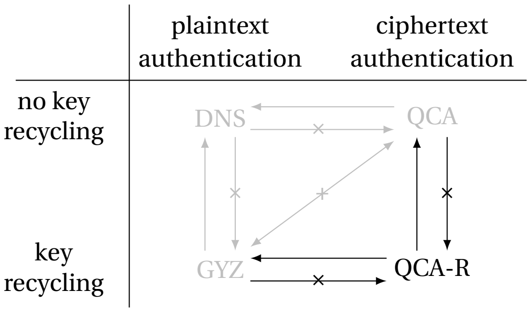


Figure 3.1: Overview of different definitions of quantum authentication. Three previously defined notions (in gray) and their relations were already known: DNS [DNS12] is strictly weaker than GYZ [GYZ17] (total authentication) and QCA [AGM18]. These last two are incomparable: there exist schemes that satisfy either one, but not the other. On the bottom right, our new definition QCA-R is displayed: it is strictly stronger than both GYZ and QCA.

where $\mathcal{S} := \mathcal{S}_{\mathsf{acc}} + \mathcal{S}_{\mathsf{rej}}$ is trace preserving, and $\mathcal{S}_{\mathsf{acc}}$ is as in Definition 3.2.3, i.e.,

$$\mathcal{S}_{\mathsf{acc}} : \rho^R \mapsto \mathop{\mathbb{E}}_{k',r} \left[\langle \varphi_{k',r}|^T \langle \Phi^+|^{M_1M_2} U_{k'}^\dagger \mathcal{A}^{M_1TR} \left(U_{k'}^{M_1T} \rho_{k',r}^{RM_1M_2T} U_{k'}^\dagger \right) U_{k'} |\varphi_{k',r}\rangle |\Phi^+\rangle \right]$$

for $\rho_{k',r} := \rho^R \otimes |\Phi^+\rangle\langle\Phi^+|^{M_1M_2} \otimes |\varphi_{k',r}\rangle\langle\varphi_{k',r}|^T$.

The first condition (closeness of the real and ideal channel) is a strengthening of Definition 3.2.2: following Portmann [Por17], we also consider which part of the key can be recycled in the reject case. If the recycling function f is the identity function, all of the key can be recycled. If f maps all keys to the all-zero string, then no constraints are put on key leakage in the reject case.

QCA-R strengthens both GYZ and QCA, but not vice versa: the codes sketched above that separate GYZ from QCA are immediately examples of codes that are GYZ (or QCA) but cannot be QCA-R. See Figure 3.1.

3.2.2 Quantum error-correcting codes

An $[[n,m]]$ quantum error-correcting code (QECC), characterized by a unitary operator V, encodes a message ρ consisting of m qubits into a codeword $V(\rho \otimes |0^t\rangle\langle 0^t|)V^\dagger$ consisting of n qubits, by appending $t := n - m$ tags $|0\rangle\langle 0|$, and applying the unitary V. Decoding happens by undoing the unitary V, and measuring the tag register in the computational basis. The measurement outcome is called the syndrome: an all-zero syndrome indicates that no error-correction is necessary. In this work, we will only use the error-detection property of QECCs, and will not worry about how to correct the message if a nonzero syndrome is measured. If that happens, we will simply discard the message (i.e., reject).

For any bit string $x \in \{0,1\}^m$, let $|x_L\rangle$ (for "logical $|x\rangle$") denote a valid encoding of $|x\rangle$, i.e., a state that will decode to $|x\rangle$ without error. A defining feature of any QECC is its distance: the amount of bit and/or phase flips required to turn one valid codeword into another. If we want to be explicit about the distance d of an $[[n,m]]$ code, we will refer to it as an $[[n,m,d]]$ code.

Definition 3.2.5 (Distance)**.** The *distance* of an $[[n,m]]$ code is the minimum weight[1] of a Pauli P such that $P|x_L\rangle = |y_L\rangle$ for some $x \neq y$, with $x, y \in \{0,1\}^m$.

An important class of quantum error-correcting codes is formed by the *CSS codes* [CS96; Ste96]. A CSS code is constructed from two classical codes, C_1 and $C_2 \subseteq C_1$. If C_1 is an $[n,m_1]$ code (encoding m bits into a codeword of length n), C_2 is an $[n,m_2]$ code, and the distance of C_1 and the dual $C_2^\perp$ is at least d, then the resulting quantum error-correcting code $CSS(C_1,C_2)$ is an $[[n, m_1 - m_2, d]]$ code, also with distance d.

The *stabilizer* of an $[[n,m,d]]$ QECC is the group of Paulis that leaves all codewords intact, i.e., the set $\{P \in \mathscr{P}_n \mid \forall x \in \{0,1\}^m : P|x_L\rangle = |x_L\rangle\}$. The stabilizers of a CSS code $CSS(C_1,C_2)$ are generated by the rows of the following table [NC00]:

$$\left[\begin{array}{c|c} H(C_2^\perp) & 0 \\ 0 & H(C_1) \end{array}\right], \tag{3.3}$$

where $H(\cdot)$ represents the parity-check matrix of the classical code. Each row in the check matrix is a concatenation of two strings x and z, from the left-hand side and the right-hand side of the dividing line, respectively. The stabilizer generator corresponding to that row is $\mathsf{X}^x\mathsf{Z}^z$.

3.2.3 (Strong) purity testing

A generic way to build quantum authentication codes is from purity-testing families of quantum error-correcting codes. In a cryptographic setting, it can be useful to select a code from a set of codes $\{V_k\}_{k\in\mathcal{K}}$ for some key set $\mathcal{K}$. We will again assume that the key k is selected uniformly at random.

Following earlier work [BCG+02; Por17], we restrict our attention to codes for which applying a Pauli to a codeword is equivalent to applying a (possibly different) Pauli directly to the message and tag register. In other words, the unitary V must be such that for any $P_\ell \in \mathscr{P}_{m+t}$, there exists a $P_{\ell'} \in \mathscr{P}_{m+t}$ and a $\theta \in \mathbb{R}$ such that $P_\ell V = e^{i\theta} V P_{\ell'}$. With our attention restricted to codes with this property, we can meaningfully define the following property:

[1]The weight of a Pauli is the number of non-identity components. See Section 2.4.2.

Definition 3.2.6 (Purity testing [BCG+02]). A family of quantum error-correcting codes $\{V_k\}_{k\in\mathcal{K}}$ is *purity testing* with error ε if for any Pauli $P_\ell \in \mathscr{P}_{m+t}\backslash\{\mathsf{I}^{\otimes(m+t)}\}$,

$$\Pr_k\left[V_k^\dagger P_\ell V_k \in (\mathscr{P}_m\backslash\{\mathsf{I}^{\otimes m}\})\otimes\{\mathsf{I},\mathsf{Z}\}^{\otimes t}\right] \leq \varepsilon.$$

In words, Definition 3.2.6 states that for any nonidentity Pauli, the probability (over the key) that the Pauli alters the message but is not detected (i.e., no tag bit is flipped) is upper bounded by ε.

Note that purity-testing codes do not necessarily detect *all* Pauli attacks with high probability: it may well be that a Pauli attack remains undetected, because it acts as identity on the message. In Section 3.3.2, we will see an example of such an attack, and how it can be used to learn information about the encoding key. This exploitation of purity-testing codes has led Portmann to consider a stronger notion of purity testing that should allow for keys to be safely reusable. In this definition, even Paulis from the set $\{\mathsf{I}^{\otimes m}\}\otimes\{\mathsf{I},\mathsf{Z}\}^{\otimes t}$, which act as identity on the message, should be detected:

Definition 3.2.7 (Strong purity testing [Por17]). A family of quantum error-correcting codes $\{V_k\}_{k\in\mathcal{K}}$ is *strong purity testing* with error ε if for any Pauli $P_\ell \in \mathscr{P}_{m+t}\backslash\{\mathsf{I}^{\otimes(m+t)}\}$,

$$\Pr_k\left[V_k^\dagger P_\ell V_k \in \mathscr{P}_m\otimes\{\mathsf{I},\mathsf{Z}\}^{\otimes t}\right] \leq \varepsilon.$$

Barnum et al. [BCG+02] described a canonical method of turning a QECC set $\{V_{k_1}\}_{k_1\in\mathcal{K}_1}$ into a symmetric-key encryption scheme. The encryption key k consists of two parts: the key $k_1\in\mathcal{K}_1$ for the QECC, and an additional one-time pad key $k_2\in\{0,1\}^{2(m+t)}$. The encryption map is then defined by setting $U_{k_1,k_2} := P_{k_2}V_{k_1}$, and $\sigma_{k_1,k_2} = |0^t\rangle\langle 0^t|$. Since σ_{k_1,k_2} is key-independent, the projectors $\Pi^{\mathsf{acc}} = |0^t\rangle\langle 0^t|$ and $\Pi^{\mathsf{rej}} = \mathbb{I} - |0^t\rangle\langle 0^t|$ are key-independent as well. In Construction 3.2.8, the complete protocol is described. Protocols of this form are also called "encode-encrypt schemes" [BGS13].

Construction 3.2.8 ([BCG+02]). Let $\{V_{k_1}\}_{k_1\in\mathcal{K}_1}$ be an $[[m+t,m]]$ quantum error-correcting code. Define the following symmetric-key encryption scheme:

Key generation: Sample $k_1 \leftarrow \mathcal{K}_1$. Sample $k_2 \leftarrow \mathcal{K}_2 := \{0,1\}^{2(m+t)}$.

Encryption: $\rho^M \mapsto P_{k_2}^{MT} V_{k_1}^{MT}\left(\rho^M \otimes |0^t\rangle\langle 0^t|^T\right) V_{k_1}^{MT} P_{k_2}^{MT}$.

Decryption: $\rho^{MT} \mapsto \langle 0^t|\left(V_{k_1}^\dagger P_{k_2}^\dagger \rho P_{k_2} V_{k_1}\right)|0^t\rangle$
$\quad + \ |\bot\rangle\langle\bot|^M \otimes \mathrm{Tr}_M\left[\sum_{i\neq 0^t}\langle i|\left(V_{k_1}^\dagger P_{k_2}^\dagger \rho P_{k_2} V_{k_1}\right)|i\rangle\right]$

When using Construction 3.2.8 with a strong-purity-testing code, plaintext authentication with key recycling is achieved:

	DNS	GYZ	QCA	QCA-R
Signed polynomial code [BCG+06]	✓	?	?	?
8-design code [GYZ17]	✓	✓	✓	✓
Clifford code [ABEM17]	✓	✓	✓	✓
Trap code [BGS13]	✓	×	✓	×
Auth-QFT-Auth [GYZ17]	✓	×	?	×
Constant-memory trap code (this work)	✓	×	✓	×
Strong trap code (this work)	✓	✓	✓	✓

Figure 3.2: Overview of quantum authentication codes and the security definitions they satisfy. For all codes except the constant-memory trap code, the security error ε is inverse exponential in the ciphertext size (provided that a sensible choice is made for the underlying CSS code for the trap code, or the underlying classical message-authentication code for the Auth-QFT-Auth code). For the constant-memory trap code, the error ε is inverse polynomial.

Lemma 3.2.9 ([Por17, Theorem 3.5]). *Let $\{V_{k_1}\}_{k_1 \in \mathcal{K}_1}$ be a strong-purity-testing code with error ε. The encryption scheme resulting from Construction 3.2.8 is quantum plaintext $\left(\sqrt{\varepsilon} + \frac{1}{2}\varepsilon\right)$-authenticating with key recycling* (GYZ). *In the reject case, the key k_1 (but not necessarily k_2) can still be recycled.*

If a code is used for the construction that is only purity-testing, the resulting encryption scheme is plaintext authenticating [BCG+02], but not necessarily with key recycling.

In Theorems 3.5.1 and 3.6.1, we will show that the analogous statements also hold for ciphertext authentication: purity-testing codes give rise to ciphertext-authenticating schemes through Construction 3.2.8, while strong-purity-testing codes additionally provide key recycling.

3.3 Codes

In this section, we take a look at a few concrete quantum authentication codes. We will analyze which variants of the authentication definitions presented in Section 3.2 they satisfy. Many codes are built on (purity-testing) quantum error-correcting codes using Construction 3.2.8, so we may immediately conclude that they provide ciphertext authentication and/or key recycling (see Lemma 3.2.9 and the text below it). Two codes do not follow Construction 3.2.8: the signed polynomial code encodes qudits rather than qubits, and the Auth-QFT-Auth is structured differently altogether. Hence, for those two codes a separate proof of their key recycling and/or ciphertext authenticating properties would be required. For an overview of the codes and their known properties, see Figure 3.2.

3.3.1 The Clifford code

A simple yet powerful code is the Clifford code [ABEM17]. For an m-qubit message, the code is specified by fixing a parameter t, setting $\sigma_k = |0^t\rangle\langle 0^t|$ for all k, and $U_k = C \in_R \mathscr{C}_{t+m}$, where the key k determines the random selection of C. Concretely, a ciphertext for an m-qubit message $|\psi\rangle$ is of the form

$$C\left(|\psi\rangle \otimes |0^t\rangle\right), \tag{3.4}$$

where C is a uniformly random Clifford. Note that, from the point of view of someone who does not know the key k, the encoding of the Clifford code looks like a Clifford twirl (see Section 2.4.6) of the input state plus some trap states.

We can interpret the Clifford code either as an authentication code $\{(U_k, \sigma_k)\}_k$ as described above, or, alternatively, as a family of quantum error-correcting codes $\{V_k\}$ with each V_k being a Clifford group element. Taking on this second point of view, and applying Construction 3.2.8 to the set of codes $\{V_k\}$, we recover exactly the code $\{(U_k, \sigma_k)\}_k$: the additional quantum one-time pad, consisting of a random Pauli, is absorbed into the random Clifford.

Viewing the Clifford code as a family of quantum error-correcting codes, we can show the following property (which in fact still holds if the Clifford group is replaced by any other unitary 2-design [Por17]).

Lemma 3.3.1. *The Clifford code on m qubits with tag-register size t is strong purity testing with error $O(2^{-t})$.*

Proof. The statement follows from the fact that random Clifford group elements map nonidentity Paulis to random nonidentity Paulis by conjugation, as stated in Corollary 2.4.2. Only a small fraction of those random nonidentity Paulis will leave the computational basis of all tag states intact.

More formally, for all nonidentity Pauli operators $P_\ell \in \mathscr{P}_{m+t} \setminus \{\mathsf{I}^{\otimes(m+t)}\}$,

$$\begin{aligned}
\Pr_{C \in \mathscr{C}_{m+t}}\left[C^\dagger P_\ell C \in \mathscr{P}_m \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes t}\right] &= \sum_{Q \in \mathscr{P}_m \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes t}} \Pr_{C \in \mathscr{C}_{m+t}}\left[C^\dagger P_\ell C = Q\right] && (3.5)\\
&= \sum_{Q \in \mathscr{P}_m \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes t}} \frac{|\{C \in \mathscr{C}_{m+t} \mid C P_\ell C = Q\}|}{|\mathscr{C}_{m+t}|} && (3.6)\\
&\overset{\text{Cor. 2.4.2}}{=} \sum_{Q \in \mathscr{P}_m \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes t}} \frac{1}{|\mathscr{P}_{m+t}| - 1} && (3.7)\\
&= \frac{|\mathscr{P}_m| \cdot 2^t}{|\mathscr{P}_{m+t}| - 1} && (3.8)\\
&= O(2^{-t}). && (3.9)
\end{aligned}$$

□

Note that the security error of the Clifford code does not depend on the message size m. For simplicity, we will mostly work with messages of size $m = 1$, but in principle, much larger messages can be securely authenticated in the Clifford code.

From the fact that it is strong purity testing, we can conclude that the Clifford code is plaintext $O(2^{-t})$-authenticating (DNS). It was already shown to have this property directly [ABEM17; AM17], combining Corollary 2.4.2 with the fact that the Clifford twirl decomposes unitaries into Pauli mixtures (Lemma 2.4.8). For completeness, we restate the proof here to establish the exact form of the simulator maps $\mathcal{S}_{\mathsf{acc}}$ and $\mathcal{S}_{\mathsf{rej}}$.

Lemma 3.3.2 (Variation on [AM17, Theorem 3.7])**.** *Let U^{MTR} be a unitary, and write $U = \sum_{x,z\in\{0,1\}^{m+t}} (\mathsf{X}^x\mathsf{Z}^z)^{MT} \otimes U^R_{x,z}$. Then for any state ρ^{MR},*

$$\Bigg\| \mathcal{T}^{MT}_{\mathscr{C}_{m+t}}(U)\left(\rho^{MR}\otimes |0^t\rangle\langle 0^t|^T\right) - \left(U^R_{0,0}\rho U^\dagger_{0,0}\otimes|0^t\rangle\langle 0^t|^T + \mathrm{Tr}_M\left[\sum_{(x,z)\neq(0,0)} U^R_{x,z}\rho U^\dagger_{x,z}\right]\otimes\tau^{MT}\right)\Bigg\|_{\mathrm{tr}} \leq \mathrm{negl}(t).$$

Proof. Using the Clifford twirl in the first step, and writing $U = \sum_{x,z}(\mathsf{X}^x\mathsf{Z}^z)^{MT}\otimes U^R_{x,z}$, we derive

$$\begin{aligned}
&\mathcal{T}^{MT}_{\mathscr{C}_{m+t}}(U)\left(\rho\otimes|0^t\rangle\langle 0^t|\right)\\
&= \sum_{x,z}\mathop{\mathbb{E}}_{C}(C\mathsf{X}^x\mathsf{Z}^zC^\dagger\otimes U_{x,z})\left(\rho\otimes|0^t\rangle\langle 0^t|\right)(C^\dagger\mathsf{X}^x\mathsf{Z}^zC\otimes U^\dagger_{x,z})\\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)}\mathop{\mathbb{E}}_{C}(C\mathsf{X}^x\mathsf{Z}^zC^\dagger\otimes U_{x,z})\left(\rho\otimes|0^t\rangle\langle 0^t|\right)(C^\dagger\mathsf{X}^x\mathsf{Z}^zC\otimes U^\dagger_{x,z}) \qquad (3.10)\\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)}\mathop{\mathbb{E}}_{(x',z')\neq(0,0)}(\mathsf{X}^{x'}\mathsf{Z}^{z'}\otimes U_{x,z})\left(\rho\otimes|0^t\rangle\langle 0^t|\right)(\mathsf{X}^{x'}\mathsf{Z}^{z'}\otimes U^\dagger_{x,z}) \qquad (3.11)\\
&\approx_{\mathrm{negl}(t)} U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)} U^R_{x,z}\left(\mathcal{T}^{MT}_{\mathscr{P}_{m+t}}\left(\rho\otimes|0^t\rangle\langle 0^t|\right)\right)U^\dagger_{x,z}\\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \tau^{MT}\otimes\sum_{(x,z)\neq(0,0)}\mathrm{Tr}_M\left[U^R_{x,z}\rho U^\dagger_{x,z}\right].
\end{aligned}$$

In the step from Equation (3.10) to Equation (3.11), we used the fact that any nonidentity Pauli is mapped to a random nonidentity Pauli by expectation over the Clifford group (Corollary 2.4.2). The approximation is due to the fact that the Pauli given by $(x', z') = (0, 0)$ needs to be included into the expectation in order to form the Pauli twirl. □

Corollary 3.3.3. *The Clifford authentication code with t trap qubits is* $\mathrm{negl}(t)$-DNS.

Proof. In the decoding procedure for the t-trap Clifford code, the register T is measured using the two-outcome measurement defined by the projector $\Pi := |0^t\rangle\langle 0^t|$. Note that, given an attack $\mathcal{A}$,

$$\mathop{\mathbb{E}}_{k\in\mathcal{K}}\left[\mathsf{Dec}_k\left(\mathcal{A}\left(\mathsf{Enc}_k\left(\rho\right)\right)\right)\right] = \mathcal{L}^{\Pi}\left(\mathcal{T}_{\mathscr{C}_{m+t}}(\mathcal{A})\left(\rho^{MR}\otimes|0^t\rangle\langle 0^t|^{T}\right)\right),$$

where $\mathcal{L}^{\Pi}(X) := \mathrm{Tr}_T[\Pi X\Pi] + |\bot\rangle\langle\bot|^M \otimes \mathrm{Tr}_{MT}[\overline{\Pi}X\overline{\Pi}]$. Then apply Lemma 3.3.2, and use the fact that $\mathrm{Tr}[|0\rangle\langle 0|^T\,\tau^{MT}] = 2^{-t}$. In the terminology of Definition 3.2.1, we may explicitly describe $\mathcal{S}_{\mathsf{acc}} := U_{0,0}(\cdot)U_{0,0}^{\dagger}$ and $\mathcal{S}_{\mathsf{rej}} := \sum_{(x,z)\neq(0,0)} U_{x,z}(\cdot)U_{x,z}^{\dagger}$ for $\mathcal{A} = U(\cdot)U^{\dagger}$ and U decomposed as in the statement of Lemma 3.3.2. In words, the only part of the attack map that will be accepted is the identity Pauli $P_{0,0}$ on the message and tag registers. □

The Clifford code fulfills all security definitions from Section 3.2.1. That is, in addition to DNS, it also allows key recycling (GYZ), fulfills QCA [AGM18], and, as we will see in Section 3.6, even QCA-R.

So why would one want to look further? A possible drawback in a client-server setting is that encoding and decoding of the Clifford code require the client to be able to perform *arbitrary* Clifford group operations. If the goal is to minimize the required quantum capabilities of the client, we may want to look for simpler codes that have similar security guarantees.

3.3.2 The trap code

An example of an authentication code with a simpler encoding procedure is the trap code [BGS13]. This scheme encrypts single-qubit messages by applying a fixed $[[n,1,d]]$ CSS code E to the message, which produces n physical qubits, and then appending $2n$ "trap" qubits (n computational-basis traps in the state $|0\rangle\langle 0|$, and n Hadamard-basis traps in the state $|+\rangle\langle +|$). The resulting $3n$ qubits are permuted in a random fashion according to a key k_1, and one-time padded with a second key k_2. At decryption, the one-time pad and permutation are removed, the traps are measured in their respective bases, and the syndrome of the CSS code is checked.[2] The trap code, for a key $k = (k_1, k_2)$, is characterized by $U_k = P_{k_2}\pi_{k_1}(E\otimes \mathsf{I}^{\otimes n}\otimes \mathsf{H}^{\otimes n})$ and $\sigma_k = |0\rangle\langle 0|^{\otimes(3n-1)}$, where π_{k_1} is a unitary that permutes the $3n$ qubits. See Figure 3.3.

We may view the first part of the trap code, without the one-time pad P_{k_2}, as a set of error-correcting codes. In this framework, the trap code is described as a QECC family $\{V_{k_1}\}$ with $m = 1$, $t = 3n-1$, and $V_{k_1} = \pi_{k_1}(E\otimes \mathsf{I}^{\otimes n}\otimes \mathsf{H}^{\otimes n})$. This code is purity testing with error $\varepsilon = (2/3)^{d/2}$ [BGS13]. Intuitively, an attack Pauli P_ℓ can only act nontrivially on the data if its weight is at least d: otherwise, it will be detected by the

[2] We differ from the analysis by Broadbent and Wainewright [BW16] in that we consider the variant that uses error detection instead of error correction on the data qubits.

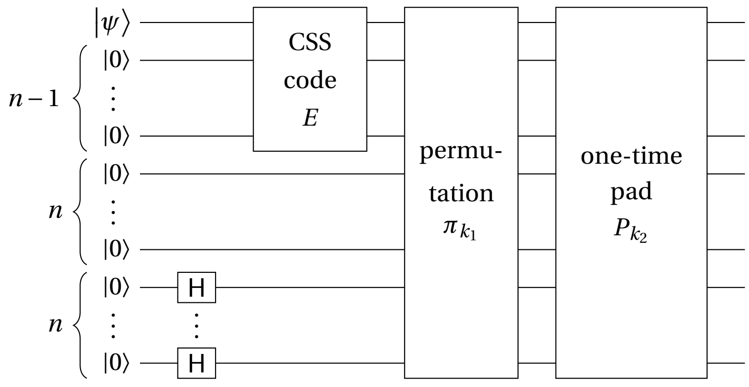


Figure 3.3: The encoding circuit for the trap code. The CSS code E maps a single logical qubit $|\psi\rangle$ to n physical qubits, using $n-1$ auxiliary qubits in the state $|0\rangle$ as input.

CSS code E. However, since the permutation is random and unknown by the attacker, a high-weight attack P_ℓ has a significant probability of landing on a trap. The proof idea [BGS13; BW16] is similar to the proof of Lemma 3.3.4 below. As we will show in Section 3.6, the purity testing property causes the trap code to satisfy QCA as well.

The trap code is *not* strong purity testing for subconstant ε. To see this, consider the Pauli $P = \mathsf{X} \otimes \mathsf{I}^{\otimes(m+t-1)}$: with probability $2/3$, the X is permuted onto a data qubit or a $|0\rangle\langle 0|$ trap, and is detected. With probability $1/3$, it hits a $|+\rangle\langle +|$ trap and remains undetected, that is, it acts as $\mathsf{H}^\dagger\mathsf{X}\mathsf{H} = \mathsf{Z}$ on one of the $2n$ tag qubits. P forms a counterexample to the strong-purity-testing property that must hold for all Paulis.

An attacker may use the above strategy to learn information about the permutation π_{k_1}. Applying the attack P, and observing whether or not the verification accepts, the attacker can learn whether or not π_{k_1} permutes a $|+\rangle\langle +|$ trap into the first position. Repeating this attack at different locations in the ciphertext, and also with single-qubit Z Paulis, an attacker can learn all locations and types of traps in at most $2(m+t)$ rounds. For this reason, the trap code is also not plaintext ε-authenticating with key recycling for sub-constant ε. It is not even clear whether the trap code can be regarded as a scheme with *partial* key leakage [GYZ17], because of the adaptive way in which it can be attacked.

3.3.3 The constant-memory trap code (variation)

In its original definition, exactly half of the $2n$ traps are designated as computational-basis traps, and half as Hadamard-basis traps. Similar security guarantees can be achieved if each trap is independently chosen to be a computational-basis trap (with probability $1/2$), or a Hadamard-basis trap (with probability $1/2$). On expectation, n traps of each type will be present. Assigning the traps in this way requires a bigger secret key, because the type of each of the $2n$ traps needs to be stored. However, it can in some cases simplify the analysis of the trap code.

In this section, we will build a variant of the trap code (the "constant-memory trap code") that assigns traps in this way [AG19]. We stress, however, that this change is not unique to this variant of the trap code. One could easily define the regular trap code with randomly assigned traps, or the constant-memory trap code with fixed traps.

The code presented in this section requires only a constant-size quantum memory from the encoder and decoder. Its security guarantee is weaker than that of the regular trap code: the purity-testing error is inverse polynomial in n, rather than inverse exponential. However, for near-term applications in delegated quantum computation this trade-off between security and client memory size may be worthwhile.

The constant-memory trap code is very similar to the regular trap code, except we use a CSS code E which maps a single logical qubit to a constant number of physical qubits c. A client can perform the trap-code encoding of this variant using a quantum memory of size $c+1$, as follows. First, he computes the encoded data state $E\left(|\psi\rangle \otimes |0^{c-1}\rangle\right)$. Then, he classically samples keys k_1, k_2, and k_3 (the third key contains $2n$ bits, and is used to determine the type of trap). He can now send the encoded state to a server qubit-by-qubit: whenever the permutation key k_1 dictates that a data qubit should be sent, it is retrieved from the first c bits of quantum storage, the appropriate one-time pad is applied using key k_2, and the qubit is sent off. Whenever a trap qubit is supposed to be sent, it can be generated independently using the last space in the quantum memory (and keys k_2 and k_3), and immediately be sent.

Lemma 3.3.4. *The constant-memory trap code, when based on a* $[[c,1,d]]$ *CSS code for constants c and d, is* $O(n^{-d/2})$*-purity testing.*

Proof. Consider a Pauli $P_\ell \in \mathscr{P}_{2n+c} \setminus \{\mathsf{I}^{\otimes(2n+c)}\}$. We consider two possible cases for the weight w of P_ℓ: either $w > d\log n$, or $w \leqslant d\log n$. Intuitively, the first case (a "high-weight" Pauli) will cause at least one trap to be triggered with high probability, while the second case (a "low-weight" Pauli) will likely not hit enough of the c data qubits in order to alter the logical state. We now analyze the two cases in detail.

In case $w > d\log n$, the probability of *not* triggering any traps is upper bounded by this probability for the lowest-possible weight $w^* = d\log n + 1$. For big enough n, we have that $d\log n+1 > c$, and so at least $d\log n+1-c$ of the nonidentity Pauli terms will land on one of the $2n$ traps qubits. Each term that lands on a trap has probability at least $1/2$ of being detected, since each trap is a random choice between a computational or Hadamard trap (and an X or Y is detected if it lands on a computational trap, while a Z or Y will be detected on a Hadamard trap). In total, the probability of *not* being detected is upper bounded by

$$\frac{1}{2^{d\log n+1-c}} = 2^{c-1}\cdot\frac{1}{2^{\log n^d}} = O(n^{-d}). \tag{3.12}$$

In case $w \leqslant d \log n$, the analysis is somewhat more involved. We are interested in an upper bound on the probability that *at least d* of the Pauli terms land on the c data qubits, which is required to act nontrivially on the data. Again, a first step is to notice that this probability is upper bounded by the extreme case $w^* = d \log n$.

For big enough n, we have that $n > d \log n$. We then analyze the probability as a sum over the possible number j of Pauli terms that land on the c data qubits:

$$\begin{aligned}
\Pr[P_\ell \text{ hits at least } d \text{ data qubits}] &= \sum_{j=d}^{c} \Pr[P_\ell \text{ hits at exactly } j \text{ data qubits}] && (3.13)\\
&= \sum_{j=d}^{c} \frac{\binom{c}{j}\binom{2n}{d\log n - j}}{\binom{2n+c}{d \log n}} && (3.14)\\
&\leqslant \sum_{j=d}^{c} \frac{\binom{c}{j}\binom{2n}{d\log n - d}}{\binom{2n+c}{d \log n}} \qquad \text{(since } d\log n < {}^{2n}/_{2}\text{)} && (3.15)\\
&= a \cdot \frac{\binom{2n}{d\log n - d}}{\binom{2n+c}{d \log n}}\,, && (3.16)
\end{aligned}$$

where a is some constant independent of n. Now, writing out the binomials and rearranging terms, we continue the derivation as

$$\begin{aligned}
\text{Eq. (3.16)} &= a \cdot \frac{(2n)!}{(2n+c)!} \cdot \frac{(d\log n)!}{(d\log n - d)!} \cdot \frac{(2n - d\log n + c)!}{(2n - d\log n + d)!} && (3.17)\\
&\leqslant a \cdot \frac{1}{(2n)^c} \cdot (d \log n)^d \cdot (2n - d\log n + c)^{c-d} && (3.18)\\
&= a \cdot \left(\frac{d\log n}{2n}\right)^d \cdot \left(\frac{2n - d\log n + c}{2n}\right)^{c-d}. && (3.19)
\end{aligned}$$

Going from Equation (3.17) to Equation (3.18), we used the fact that for $x \geqslant y$, $(x!)/(y!) = (y+1)(y+2)\cdots(x-1)x \leqslant x^{x-y}$. If n is big enough such that $c < d \log n$, the right-most term in the product in Equation (3.19) is smaller than 1. Furthermore using the fact that $\log n = O(\sqrt{n})$, we can upper bound Equation (3.19) as

$$O\left(\left(\frac{d\sqrt{n}}{2n}\right)^d\right) = O\left(\left(\frac{1}{\sqrt{n}}\right)^d\right) = O\left(n^{-d/2}\right). \qquad (3.20)$$

We conclude that in both cases ($w > d\log n$ and $w \leqslant d \log n$), the probability that P_ℓ acts as identity on the data qubit and at the same time triggers no traps, is inverse polynomial as $O(n^{-d/2})$. □

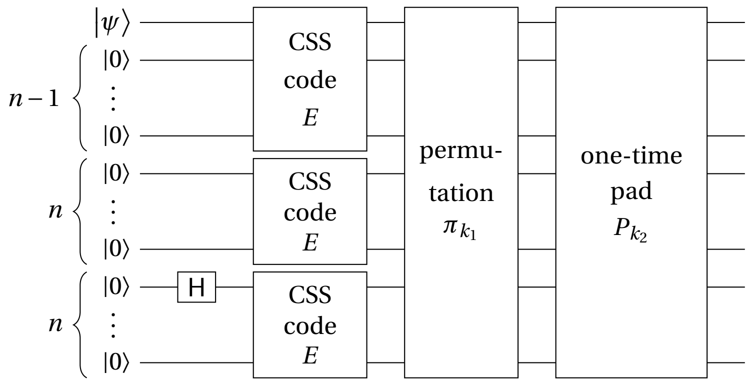


Figure 3.4: The encoding circuit for the strong trap code. Each of the three qubits $|\psi\rangle$ (the data), $|0\rangle$ (the computational-basis trap), and $|+\rangle$ (the Hadamard-basis trap) are encoded under E. This contrasts the regular trap code, which only encodes the data qubit under a CSS code.

3.3.4 The strong trap code (variation)

We present another modified version of the trap code, which we call the *strong trap code*. Contrary to the regular trap code, which appends $2t$ trap qubits, the strong trap code only appends a single $|0\rangle$ trap and a single $|+\rangle$ trap. These two traps are subsequently encoded using a quantum error-correcting code, resulting in a ciphertext of the same length as the original trap code. See Figure 3.4.

The strong trap code invokes two layers of security: the CSS codes E, which detect low-weight attacks, and the traps $|0\rangle$ and $|+\rangle$, which detect higher-weight attacks by revealing bit and phase flips, respectively. Because of this double layer of security, an attack such as described in Section 3.3.2 will not be successful.

In Section 3.7, we will show that the strong trap code is strong purity testing, provided that the underlying CSS code E is chosen correctly. In particular, the stabilizers for E should not have low weight, and should not be concentrated around one particular weight. In Section 3.7, we provide details on the properties the CSS code should satisfy, construct an explicit code that satisfies them, and give a formal specification of the strong trap code.

3.3.5 Other codes

For a complete exposition, we briefly mention several other quantum authentication codes that have appeared in the literature. They do not play a prominent role in the rest of this work.

The (signed) polynomial code

The polynomial code [AB08] is a quantum error-correcting code that encodes general qudits (i.e., quantum states in a higher-dimensional Hilbert space) instead of qubits. It employs the CSS construction (see Section 3.2.2) on two classical codes C_1 and $C_2^{\perp}$. Both C_1 and C_2 consist of lists of $2d+1$ points from degree-d polynomials. Since such polynomials are fixed by any set of $d+1$ points, if one changes d or fewer points in the list, the resulting list is only consistent with polynomials of degree greater than d (and therefore cannot be a valid codeword for C_1 or C_2). This principle makes the polynomial code a distance-d QECC.

The *signed* polynomial code [BCG+06] is an adaptation of the polynomial code into an authentication code. Every element in the list of points is multiplied by a secret value $k_i \in \{-1,1\}$, independently sampled for every $1 \leqslant i \leqslant 2d+1$. Additionally, a random Pauli operator (generalized for qudits) is applied to the entire state. As such, we can view the signed polynomial code as a family $\{V_k\}_{k\in\{-1,1\}^{2d+1}}$ of encoding unitaries V_k, followed by a quantum one-time pad, very much akin to Construction 3.2.8.

The signed polynomial code is strong purity testing [ABEM17; Por17], but since it acts on qudits rather than qubits, plaintext authentication with key recycling does not immediately follow. Portmann's proof [Por17] would have to be generalized from qubits to qudits. The polynomial code *is* separately proven to be (2^{-d})-plaintext authenticating [ABEM17].

Encoding of the signed polynomial code requires a Fourier transform (which is a generalized Hadamard), a multiplication gate (which does not have an analogue in the qubit setting), and several SUM gates (which are generalized CNOT gates). This encoding procedure is arguably less demanding than the Clifford code, but an objective comparison between this set of qudit operations and the qubit Clifford group heavily depends on the architecture underlying the quantum computer.

The Auth-QFT-Auth code

Essentially, a quantum authentication code needs to authenticate both the information in the computational basis, and the information in the Hadamard basis. A code that explicitly does so is the Auth-QFT-Auth code [GYZ17]: it applies a classical message-authentication code in the computational basis (the "inner encoding"), performs a quantum Fourier transform to shift the state to the Hadamard basis, and then applies the classical message-authentication code again (the "outer encoding").

The Auth-QFT-Auth code does not follow the structure of Construction 3.2.8, so nothing can be said about the purity testing of some underlying quantum error-correcting code. It can be directly shown [GYZ17] that the Auth-QFT-Auth code is plaintext ε-authenticating where ε depends on the security of the underlying classical code. The key of the inner authentication can be recycled in the accept case, but the key of the outer authentication cannot. Therefore, it does not provide key recycling in the sense of definition Definition 3.2.2.

A code based on unitary 8-designs

The first known code to allow full key recycling was presented by Garg, Yuen and Zhandry [GYZ17]: its structure is exactly like the Clifford code, except that instead of the Clifford group it requires a more complicated group of unitaries that forms a so-called 8-design. Later, it was shown that any 2-design (and in particular the Clifford group) actually suffices for key recycling [Por17]. The 8-design code satisfies the same security properties as the Clifford code, but is not known to be strictly stronger in any sense. Therefore, the Clifford code is preferable due to the relative simplicity of its encoding and decoding, compared to an 8-design.

3.4 Quantum computing on authenticated data

The central role of (quantum) authentication codes is to ensure that a message remains unaltered while it is sent over a communication channel or stored at an untrusted location. Their applications are much wider than that, however: some quantum authentication codes have the option to allow only a very *specific* (set of) computation(s) on the plaintext message. The idea of quantum computing on authenticated data (QCAD) has found applications in multi-party quantum computation [BCG+06] (see also Chapter 4), interactive delegated quantum computation [ABEM17], one-time programs [BGS13], and quantum homomorphic encryption (see Chapter 5).

In this section, we explain the ideas behind quantum computing on data that is encoded in the Clifford code or the trap code. Both procedures will be used later on in this work. Computing on quantum states authenticated with the strong trap code works in much the same way as for the original trap code.

For both codes, the setup is as follows. There is a *client* (the delegating party) that generates the key and encrypts some input state ρ. The other party is the *server* (the computing party, sometimes called the *prover*), who receives the encoded state (but not the key!), and is supposed to perform some circuit D on the logical level. That is, the server sends a state back to the client that should decode to $\Phi_D(\rho)$, if the client uses an updated key k', that is a function of the original key k, the circuit D, and possibly some extra information generated by the server. The server achieves this by applying a procedure Ψ_D so that

$$\mathsf{Dec}_{k'}\left(\Psi_D\left(\mathsf{Enc}_k(\rho)\right)\right) = \Phi_D(\rho). \tag{3.21}$$

Generally, Ψ_D is a concatenation of subprocedures for the gates in D. Similarly to Definition 3.2.1, for any action of the server, the decoding will only accept if the output state is close to the correct output $\Phi_D(\rho)$ in trace norm.

The circuit D is public, and known to both parties. Depending on the application, we can think of D as a fixed circuit, a circuit chosen by the client (and communicated to the server), or vice versa.

3.4.1 On the Clifford code

Basic quantum computation on Clifford-authenticated data is fairly straightforward, since the encoding/decoding procedures are very powerful. In fact, all of the Clifford computational tasks will de facto be performed by the client. This also means that the Clifford code is not very well suited for *outsourcing* computations from a client to a server.

Suppose that D is a single-qubit Clifford unitary, to be applied to a Clifford-encoded state of the form $C\left(\rho\otimes|0^t\rangle\langle 0^t|\right)C^\dagger$. Then we can achieve QCAD by setting $\Psi_D = \mathbb{I}$, and defining the decryption key C' to be $C' := C(D^\dagger \otimes \mathsf{I}^{\otimes t})$. Then the state after the decrypting Clifford (but right before the measurement of the traps) is

$$\left(D\otimes\mathsf{I}^{\otimes t}\right)C^\dagger C\left(\rho\otimes|0^t\rangle\langle 0^t|\right)C^\dagger C\left(D^\dagger\otimes\mathsf{I}^{\otimes t}\right) = D\rho D^\dagger\otimes|0^t\rangle\langle 0^t|. \qquad (3.22)$$

Thus, updating the key in this way effectively applies the unitary D to the logical state.

The above strategy, if used on multiple qubits (and a circuit D with multiple wires) *entangles* the decoding keys for the individual qubits. For example, if $D = \mathsf{CNOT}$, the decoding key for two qubits (with original keys C_1 and C_2) becomes $(C_1\otimes C_2)(\mathsf{CNOT}_{1,t+2}\otimes\mathsf{I}^{\otimes 2t})$, which itself cannot be written as a product of two keys $C_1'\otimes C_2'$. Thus, the entire output state has to be decoded at once, rather than qubit-by-qubit.

When going beyond Clifford computation, computing on Clifford-encoded data becomes significantly more challenging. This is due to the fact that the Clifford code has very little structure, apart from its own group structure. Thus, there is very little that the server (who does not know the encoding Clifford) can do without effectively performing a random action.

The simplest way to implement a computational-basis measurement is for the client to perform it himself, right after decoding. Without loss of generality, we may assume that the circuit D does not perform any gates on any measured wires, and only uses them to *classically control* Clifford gates on other wires. Since Clifford gates are performed by the client updating his key, he can choose to only perform those key update if the measurement outcome was 1.

With procedures for Clifford computation and computational-basis measurement in place, performing a T gate can be done using magic-state computation (see Section 2.4.4). At encoding time, the client supplies the server with a sufficient number of *encoded* magic states $\mathsf{T}|+\rangle$ (one for each T gate in the circuit D). The server can then perform the *encoded* form Ψ_M of the magic-state-computation circuit M, since M consists of only Cliffords and measurements.

It is clear that QCAD using the Clifford code is generally not suitable for outsourcing computations from a less powerful client to a more powerful server. In practice, the client would still have to be able to perform all Clifford gates, measurements, and generate magic states for the T gates.[3] However, this type of computation has found an application in the slightly different setting of multi-party quantum computation, where different players outsource parts of the computation to other (equally powerful) players. Chapter 4 will describe the use of the Clifford code in this setting in more detail. In particular, it describes how measurements can be outsourced from one player to another in a more intricate way than described above.

3.4.2 On the trap code

The trap code distinguishes itself from the Clifford code in two ways. First, individually-authenticated input qubits can be entangled during the computation, but still be de-authenticated individually. In contrast, de-authentication in the Clifford code needs to happen simultaneously on all qubits that were involved in the computation, including any auxiliary ones. Second, the trap code allows for "authenticated measurements": if a third party measures a ciphertext, the client can verify the authenticity of the result from the classical measurement outcomes only. That is, the client does not have to physically perform the measurement himself.

Combined with the fact that encoding/decoding the trap code does not require the client to execute arbitrary Clifford operations, the above properties make the trap code more suitable for outsourcing computations from a less powerful client to a more powerful server, as compared to the Clifford code. Broadbent, Gutoski, and Stebila describe in detail how a server can compute on trap-code-encoded quantum states [BGS13]. Variations appear (sometimes implicitly) in other work [SP00; BJSW16]. We give an intuitive overview here, without providing proofs of correctness and security. In Sections 5.5, 5.6 and 6.4, we will apply these techniques in the context of quantum homomorphic encryption, ensuring verifiability for the client.

Quantum computing on the trap code relies on two different concepts: transversal computation and magic-state computation. Simpler gates (Pauli, CNOT and measurement) can be computed *transversally*, i.e., the gate is applied to the logical quantum state by applying it to each physical qubit individually. Other gates (H, P, and T) are applied using magic-state computation, similarly to Section 3.4.1.

Pauli gates. Suppose the server intends to apply an X gate to the encoded single-qubit state ρ,

$$P_{k_2} V_{k_1} \left(\rho \otimes |0^{3n-1}\rangle\langle 0^{3n-1}|\right) V_{k_1}^{\dagger} P_{k_2}^{\dagger}. \tag{3.23}$$

[3]Depending on the architecture, the latter may still be easier to achieve than performing T gates, since it would be possible to generate the magic states using multiple attempts, discarding magic states that are too noisy, and distilling better-quality magic states from the less noisy ones.

(Recall that $V_{k_1} = \pi_{k_1}(E \otimes \mathsf{I}^{\otimes n} \otimes \mathsf{H}^{\otimes n})$.) The simplest way is for the server to do nothing, and for the client to update the key k_2 such that $P_{k_2'} = P_{k_2}\pi_{k_1}(\mathsf{X}^{\otimes n} \otimes \mathsf{I}^{\otimes 2n})\pi_{k_1}^{-1}$: that is, all n data qubits (and none of the traps) receive an extra X gate. Since E is a CSS code, it allows transversal Pauli application, i.e., $\mathsf{X}^{\otimes n}E(|\psi\rangle \otimes |0^{n-1}\rangle) = E(\mathsf{X}|\psi\rangle \otimes |0^{n-1}\rangle)$ for all single-qubit states $|\psi\rangle$. Thus, decoding the state in Equation (3.23) with keys (k_1, k_2') results in the state $\mathsf{X}\rho\mathsf{X}^\dagger$. The procedure for Z is similar.

CNOT gates. Suppose the server intends to apply a CNOT gate to the logical two-qubit state ρ^{AB}, which is encoded as

$$\left(P_{k_{A,2}}^{AT_A}V_{k_{A,1}}^{AT_A} \otimes P_{k_{B,2}}^{BT_B}V_{k_{B,1}}^{BT_B}\right)\left(\rho^{AB} \otimes |0^{6n-2}\rangle\langle 0^{6n-2}|^{T_AT_B}\right)\left(P_{k_{A,2}}V_{k_{A,1}} \otimes P_{k_{B,2}}V_{k_{B,1}}\right)^\dagger. \tag{3.24}$$

For the encoded CNOT to work correctly, we require that $k_{A,1} = k_{B,1}$, i.e., both encodings are permuted in exactly the same way. The trap code is still secure in this case, as long as the Pauli keys $k_{A,2}$ and $k_{B,2}$ are independent [BGS13].

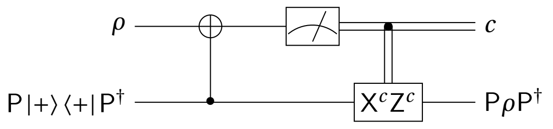
A logical CNOT is then executed by applying $3n$ physical CNOT gates to Equation (3.24), each between one qubit of AT_A, and one qubit of BT_A. The CNOT gates commute through $P_{k_{A,2}} \otimes P_{k_{B,2}}$, as long as the client updates the Pauli keys $k_{A,2}$ and $k_{B,2}$ in a fixed way (see the commutation relations between Paulis and CNOT, Equations (2.10) to (2.13)). Because the permutations are identical, the CNOT gates commute through them unchanged. Because E is a CSS code, it is transversal for CNOT: that is, $\mathsf{CNOT}^{\otimes n}E^{\otimes 2}(|\psi\rangle \otimes |0^{2n-2}\rangle) = E^{\otimes 2}(\mathsf{CNOT}|\psi\rangle \otimes |0^{2n-2}\rangle)$ for any two-qubit state $|\psi\rangle$. Finally, since $\mathsf{CNOT}|00\rangle = |00\rangle$ and $\mathsf{CNOT}(H \otimes H)|00\rangle = (H \otimes H)|00\rangle$, the CNOT operations do not affect the $2n$ traps. Thus, the final state is identical to Equation (3.24), except that ρ is now $\mathsf{CNOT}\rho\mathsf{CNOT}^\dagger$, and that the Pauli keys $k_{A,2}$ and $k_{B,2}$ have changed according to the commutation of CNOT and the Pauli group.

Computational-basis measurements. To do a computational-basis measurement, the server measures all $3n$ physical qubits in the computational basis, resulting in a measurement string $s \in \{0,1\}^{3n}$. The client, knowing the permutation, can divide the string into three n-bit strings: a data string s_{data}, a computational-trap string s_0, and a Hadamard-trap string s_+. He discards the string s_+,[4] and checks whether the string s_0 is consistent with the relevant part of the Pauli key k_2: undoing the X operations on the string s_0 should result in the all-zero string. If that is the case, the client removes the X operations on on s_{data} (again, as dictated by k_2), and runs a classical decoding procedure $\mathsf{ClassicalDec}$ on the result: it either rejects, or gives the logical measurement outcome as a single bit. The procedure $\mathsf{ClassicalDec}$ exists for

[4]The string s_+ is a fully random string, since it consists of the measurement results of $|+\rangle$ and $|-\rangle$ states. Note that measuring and discarding it means that, e.g., a logical Z attack is not detected. This does not matter, however, since such attacks do not affect the computational-basis measurement outcome.

every CSS code E [BGS13], and essentially realizes transversal measurements on the code E.

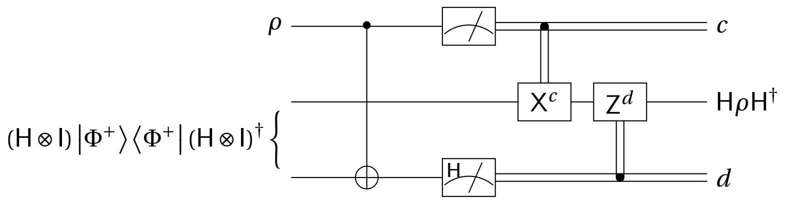
P gates. Phase gates are applied through magic-state computation: the client supplies the server with an *encoding* of the magic state $\mathsf{P}\left|+\right\rangle$, and the server runs the encoding of the magic-state computation circuit, consisting of a CNOT, computational-basis measurement, and an XZ conditioned on the measurement result:


ρ
c
$\mathsf{P}\left|+\right\rangle\left\langle+\right|\mathsf{P}^\dagger$
$\mathsf{X}^c\mathsf{Z}^c$
$\mathsf{P}\rho\mathsf{P}^\dagger$

(3.25)

Note that the client can only compute the key update for XZ after decoding the measurement result.

H gates. Hadamard gates are similar to P gates, except they require an encoding of a more complicated magic state, $(\mathsf{H}\otimes\mathsf{I})\left|\Phi^+\right\rangle$, where $\left|\Phi^+\right\rangle$ is the EPR pair. The magic-state computation performed by the server involves a CNOT, two measurements, and two Pauli corrections:


ρ
c
X^c
Z^d
$\mathsf{H}\rho\mathsf{H}^\dagger$
$(\mathsf{H}\otimes\mathsf{I})\left|\Phi^+\right\rangle\left\langle\Phi^+\right|(\mathsf{H}\otimes\mathsf{I})^\dagger$
H
d

(3.26)

One of the measurements is in the computational basis, and one is in the Hadamard basis: the server can measure in the Hadamard basis by applying H gates transversally to all qubits right before starting the procedure for a computational-basis measurement. Doing so swaps the role of the $\left|0\right\rangle$ and $\left|+\right\rangle$ traps, so the client needs to check the string s_+ against the (updated) Pauli key, instead of s_0.

The procedure for H gates requires the CSS code to be self-dual, so that Hadamard gates are indeed transversal for it [BGS13]. The concatenated Steane code is an example of a self-dual CSS code, as is the family of codes we construct in Section 3.7.3.

T gates. Finally, T gates are also similar to P gates in that they require a magic state $\mathsf{T}\left|+\right\rangle$ and a magic-state computation (see Figure 2.1). The complicating factor is that the measurement result in the magic-state computation dictates whether or not a *non-Pauli* correction PX is to be applied. The client cannot perform this correction on his own at decoding time, but needs to decode the measurement result before knowing whether to assist the server in applying PX. Broadbent et al. solve this apparent deadlock by letting client and server communicate during the protocol [BGS13]. In Chapter 6, we manage the problem in a noninteractive way, by having the client prepare a special gadget state (in addition to the magic state) before the computation.

With the procedures described above, a client holding the classical encryption key can guide a server in performing computation directly on the ciphertext. The client sends input-independent auxiliary states (the magic states) that help bypass the traps, and updates the classical keys during the computation. One can verify that computing on states authenticated with the constant-memory trap code or strong trap code works in much the same way as for the original trap code.

3.5 Purity testing implies QCA

It was already observed that if a set of quantum error-correcting codes $\{V_{k_1}\}_{k_1 \in \mathcal{K}_1}$ is purity testing, then the encryption scheme resulting from Construction 3.2.8 is plaintext authenticating [BCG+02]. We strengthen this result by showing that the construction turns purity-testing codes into *ciphertext*-authenticating schemes (see Theorem 3.5.1 below). Later, in Section 3.6, we will show that strong-purity-testing codes are similarly turned into QCA-R schemes (Theorem 3.6.1). Only purity testing is in general not enough to achieve QCA-R: the trap code is a counterexample.

Theorem 3.5.1. *Let $\{V_{k_1}\}_{k_1 \in \mathcal{K}_1}$ be a purity-testing code with error ε. The encryption scheme resulting from Construction 3.2.8 is quantum ciphertext ε-authenticating (ε-QCA).*

The proof of Theorem 3.5.1 uses ideas and techniques that are similar to Portmann's proof [Por17, Theorem 3.5], but with a simulator that runs the adversary on encrypted halves of EPR pairs, so that it is suitable for QCA. We prove that the ideal and the real channel are close by considering the accept and the reject cases separately, and by showing that they are both $\varepsilon/2$-close. First, we decompose the adversarial attack into Paulis by Pauli twirling (Lemma 2.4.5) it with the quantum-one-time-pad encryption from Construction 3.2.8. In the accept case, the difference between the real and the ideal scenario lies in those attacks that are accepted in the real case, but not in the ideal case. These are exactly those Paulis that, after conjugation with the key k_1 that indexes the purity-testing code, are in the set $(\mathscr{P}_m \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}) \backslash (\{\mathsf{I}^{\otimes m}\} \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}) = (\mathscr{P}_m \backslash \{\mathsf{I}^{\otimes m}\}) \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}$. The purity-testing property guarantees that the probability over k_1 of a Pauli attack landing in this set is small. The reject case is similar.

Proof. Let $\mathcal{A}$ be an adversary as in Definition 3.2.3. Define a simulator $\mathcal{S}$ on the side-information register R as follows: prepare an EPR pair $|\Phi^+\rangle\langle\Phi^+|$ in the register $M_1 M_2$

and encrypt the first qubit M_1 using a freshly sampled key $(k_1', k_2') \in \mathcal{K} := \mathcal{K}_1 \times \mathcal{K}_2$ (that is, initialize the tag register T in the state $|0^t\rangle\langle 0^t|$, and apply $P_{k_2'} V_{k_1'}$ to $M_1 T$). Then, run the adversary on the registers $M_1 TR$, keeping M_2 to the side. Afterwards, run the decryption procedure by undoing the encryption unitary and measuring whether the registers $M_1 M_2 T$ are in the state $|\Phi^+, 0^t\rangle\langle\Phi^+, 0^t|$ $(= |\Phi^+\rangle\langle\Phi^+| \otimes |0^t\rangle\langle 0^t|)$. If so, accept, and if not, reject. Note that this simulator is of the required form in the accept case (see Definition 3.2.3).

We show that for this simulator, the distance $\frac{1}{2}\|\mathfrak{I} - \mathfrak{R}\|_\diamond$ between the ideal and the real channel is upper bounded by ε. Let ρ^{MRE} be any quantum state on the message register, side-information register, and an environment register E. Let U^{MTR} be a unitary[5] map representing the adversarial channel $\mathcal{A}$, and let $\mu^{\text{real}}_{k_1,k_2}$ and $\mu^{\text{ideal}}_{k_1,k_2}$ be the effective output states in the real and ideal world, respectively:

$$\mu^{\text{real}}_{k_1,k_2} := V^\dagger_{k_1} P^\dagger_{k_2} U^{MTR} P^{MT}_{k_2} V^{MT}_{k_1} \left(\rho \otimes |0^t\rangle\langle 0^t|\right) V^\dagger_{k_1} P^\dagger_{k_2} U^\dagger P_{k_2} V_{k_1}, \tag{3.27}$$

$$\mu^{\text{ideal}}_{k_1,k_2} := V^\dagger_{k_1} P^\dagger_{k_2} U^{M_1TR} P^{M_1T}_{k_2} V^{M_1T}_{k_1} \left(\rho \otimes |0^t, \Phi^+\rangle\langle 0^t, \Phi^+|\right) V^\dagger_{k_1} P^\dagger_{k_2} U^\dagger P_{k_2} V_{k_1}. \tag{3.28}$$

The effective output states represent the states after the encoding, attack, and decoding (without measurement) of the input state $\rho \otimes |0^t\rangle$. The map $V^\dagger_{k_1} P^\dagger_{k_2} U P_{k_2} V_{k_2}$ is often called the *effective attack*. We can write the result of the real and the ideal channels as

$$\mathfrak{R}(\rho) = \mathop{\mathbb{E}}_{k_1,k_2}\left[\langle 0^t|^T \mu^{\text{real}}_{k_1,k_2} |0^t\rangle \; + \; |\perp\rangle\langle\perp|^M \otimes \operatorname{Tr}_M\left(\sum_{i \neq 0^t} \langle i|^T \mu^{\text{real}}_{k_1,k_2} |i\rangle\right)\right], \tag{3.29}$$

$$\mathfrak{I}(\rho) = \mathop{\mathbb{E}}_{k_1',k_2'}\left[\langle \Phi^+, 0^t|^{M_1M_2T} \mu^{\text{ideal}}_{k_1',k_2'} |\Phi^+ 0^t\rangle \; + |\perp\rangle\langle\perp|^M \otimes \operatorname{Tr}_M\left(\sum_{i \neq (\Phi^+, 0^t)} \langle i| \mu^{\text{ideal}}_{k_1',k_2'} |i\rangle\right)\right]. \tag{3.30}$$

These expressions are obtained simply by plugging in the description of the authentication scheme (see Construction 3.2.8) and the simulator into the channels of Definition 3.2.3. Since the accept states are orthogonal to the reject states in the M register, the distance $\frac{1}{2}\left\|\mathfrak{I}(\rho) - \mathfrak{R}(\rho)\right\|_{\text{tr}}$ can be written as

$$\begin{aligned}\frac{1}{2}\Bigg\| &\mathop{\mathbb{E}}_{k_1',k_2'}\left(\langle \Phi^+, 0^t| \mu^{\text{ideal}}_{k_1',k_2'} |\Phi^+, 0^t\rangle\right) - \mathop{\mathbb{E}}_{k_1,k_2} \langle 0^t| \mu^{\text{real}}_{k_1,k_2} |0^t\rangle \\ &+ \; |\perp\rangle\langle\perp|^M \otimes \left(\mathop{\mathbb{E}}_{k_1',k_2'}\left(\operatorname{Tr}_M \sum_{i \neq (0^t, \Phi^+)} \langle i| \mu^{\text{ideal}}_{k_1',k_2'} |i\rangle\right) - \mathop{\mathbb{E}}_{k_1,k_2}\left(\operatorname{Tr}_M \sum_{i \neq 0^t} \langle i| \mu^{\text{real}}_{k_1,k_2} |i\rangle\right)\right)\Bigg\|_{\text{tr}}\end{aligned} \tag{3.31}$$

[5]We can assume unitarity without loss of generality: if the adversary's actions are not unitary, we can dilate the channel into a unitary one by adding another environment and tracing it out afterwards. In the proof, the environment takes on the same role as the side-information register R, so we omit it for simplicity.

$$= \frac{1}{2} \left\| \mathop{\mathbb{E}}_{k_1',k_2'} \left(\langle \Phi^+, 0^t | \mu^{\text{ideal}}_{k_1',k_2'} | \Phi^+, 0^t \rangle \right) - \mathop{\mathbb{E}}_{k_1,k_2} \langle 0^t | \mu^{\text{real}}_{k_1,k_2} | 0^t \rangle \right\|_{\text{tr}}$$
$$+ \frac{1}{2} \left\| \mathop{\mathbb{E}}_{k_1',k_2'} \left(\mathrm{Tr}_M \sum_{i \neq (0^t, \Phi^+)} \langle i | \mu^{\text{ideal}}_{k_1',k_2'} | i \rangle \right) - \mathop{\mathbb{E}}_{k_1,k_2} \left(\mathrm{Tr}_M \sum_{i \neq 0^t} \langle i | \mu^{\text{real}}_{k_1,k_2} | i \rangle \right) \right\|_{\text{tr}}. \qquad (3.32)$$

We can thus focus on bounding the two terms in Equation (3.32), for accept and reject, separately. Intuitively, the two states inside the first trace norm in Equation (3.32) differ on those Paulis P_ℓ that are accepted in the real scenario, but not in the ideal one. The purity-testing property promises that these Paulis are very few. We will work out this case; the second (the reject case) is similar.

Decompose the attack as $U^{MTR} = \sum_\ell \alpha_\ell P_\ell^{MT} \otimes U_\ell^R$. Rewrite the real accept case as

$$\mathop{\mathbb{E}}_{k_1,k_2} \langle 0^t | \mu^{\text{real}}_{k_1,k_2} | 0^t \rangle \qquad (3.33)$$
$$= \mathop{\mathbb{E}}_{k_1,k_2} \langle 0^t | V_{k_1}^\dagger P_{k_2}^\dagger \left(\sum_\ell \alpha_\ell P_\ell^{MT} \otimes U_\ell^R \right) P_{k_2}^{MT} V_{k_1}^{MT} (\rho \otimes |0^t\rangle\langle 0^t|)$$
$$V_{k_1}^\dagger P_{k_2}^\dagger \left(\sum_{\ell'} \alpha^*_{\ell'} P_{\ell'} \otimes U_{\ell'}^\dagger \right) P_{k_2} V_{k_1} |0^t\rangle \qquad (3.34)$$
$$= \mathop{\mathbb{E}}_{k_1,k_2} \sum_{\ell,\ell'} \alpha_\ell \alpha^*_{\ell'} \langle 0^t | \left(V_{k_1}^\dagger P_{k_2}^\dagger P_\ell P_{k_2} V_{k_1} \otimes U_\ell^R \right) (\rho \otimes |0^t\rangle\langle 0^t|)$$
$$\left(V_{k_1}^\dagger P_{k_2}^\dagger P_{\ell'} P_{k_2} V_{k_1} \otimes U_{\ell'}^\dagger \right) |0^t\rangle. \qquad (3.35)$$

By the Pauli twirl lemma (Lemma 2.4.6), this last line equals

$$\mathop{\mathbb{E}}_{k_1} \sum_\ell |\alpha_\ell|^2 \langle 0^t | \left(V_{k_1}^\dagger P_\ell V_{k_1} \otimes U_\ell^R \right) (\rho \otimes |0^t\rangle\langle 0^t|) \left(V_{k_1}^\dagger P_\ell V_{k_1} \otimes U_\ell^\dagger \right) |0^t\rangle \qquad (3.36)$$
$$= \mathop{\mathbb{E}}_{k_1} \sum_{\ell : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{real}}} |\alpha_\ell|^2 \left(Q^M_{k_1,\ell} \otimes U_\ell^R \right) \rho \left(Q_{k_1,\ell} \otimes U_\ell^\dagger \right), \qquad (3.37)$$

where $Q_{k_1,\ell}$ is the effective Pauli on the message register, induced by $V_{k_1}^\dagger P_\ell V_{k_1}$, and where $\mathscr{P}_{\text{real}} := \mathscr{P}_m \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}$ is the set of (effective) Paulis that are undetected by the measurement after undoing V_{k_1} in the real scenario.

With the same techniques, we can rewrite the ideal accept case as

$$\mathop{\mathbb{E}}_{k_1',k_2'} \langle \Phi^+, 0^t | \mu^{\text{ideal}}_{k_1',k_2'} | \Phi^+, 0^t \rangle = \mathop{\mathbb{E}}_{k_1'} \sum_{\ell : V_{k_1'}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{ideal}}} |\alpha_\ell|^2 \left(Q^M_{k_1,\ell} \otimes U_\ell^R \right) \rho \left(Q^M_{k_1,\ell} \otimes U_\ell^R \right),$$
$$(3.38)$$

where $\mathscr{P}_{\text{ideal}} := \{\mathsf{I}^{\otimes m}\} \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}$ is the set of Paulis that are undetected in the ideal scenario. Note that for k_1 and ℓ such that $V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{ideal}}$, $Q_{\ell,k_1} = \mathsf{I}^{\otimes m}$.

The distance between the ideal and the real accept states is thus

$$\frac{1}{2}\left\| \mathop{\mathbb{E}}_{k'_1,k'_2}\left(\langle\Phi^+,0^t|\,\mu^{\mathsf{ideal}}_{k'_1,k'_2}\,|\Phi^+,0^t\rangle\right) - \mathop{\mathbb{E}}_{k_1,k_2}\langle 0^t|\,\mu^{\mathsf{real}}_{k_1,k_2}\,|0^t\rangle \right\|_{\mathrm{tr}} \tag{3.39}$$

$$= \frac{1}{2}\left\| \mathop{\mathbb{E}}_{k_1} \sum_{\ell : V^\dagger_{k_1} P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}} \setminus \mathscr{P}_{\mathsf{ideal}}} |\alpha_\ell|^2 \left(Q^M_{k_1,\ell}\otimes U^R_\ell\right)\rho\left(Q^M_{k_1,\ell}\otimes U^R_\ell\right)\right\|_{\mathrm{tr}} \tag{3.40}$$

$$\leqslant \frac{1}{2}\mathop{\mathbb{E}}_{k_1} \sum_{\ell : V^\dagger_{k_1} P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}} \setminus \mathscr{P}_{\mathsf{ideal}}} |\alpha_\ell|^2, \tag{3.41}$$

by the triangle inequality for the trace norm. Let $\delta_{(P_a \in A)}$ be 1 whenever $P_a \in A$, and 0 otherwise. Note that for all k_1, $V^\dagger_{k_1} P_0 V_{k_1} = \mathsf{I}^{\otimes(m+t)} \notin \mathscr{P}_{\mathsf{real}} \setminus \mathscr{P}_{\mathsf{ideal}}$, because $P_0 = \mathsf{I}^{\otimes(m+t)}$. This justifies the continuation of the derivation:

$$\text{Eq. (3.41)} = \frac{1}{2}\sum_{\ell}\mathop{\mathbb{E}}_{k_1}\delta_{(V^\dagger_{k_1}P_\ell V_{k_1}\in\mathscr{P}_{\mathsf{real}}\setminus\mathscr{P}_{\mathsf{ideal}})}|\alpha_\ell|^2 \tag{3.42}$$

$$= \frac{1}{2}\sum_{\ell\neq 0}\mathop{\mathbb{E}}_{k_1}\delta_{(V^\dagger_{k_1}P_\ell V_{k_1}\in\mathscr{P}_{\mathsf{real}}\setminus\mathscr{P}_{\mathsf{ideal}})}|\alpha_\ell|^2 \tag{3.43}$$

$$\leqslant \frac{1}{2}\sum_{\ell\neq 0}\varepsilon|\alpha_\ell|^2 \tag{3.44}$$

$$\leqslant \frac{\varepsilon}{2}. \tag{3.45}$$

The first inequality is due to the fact that $\mathscr{P}_{\mathsf{real}} \setminus \mathscr{P}_{\mathsf{ideal}} = (\mathscr{P}_m \setminus \{\mathsf{I}^{\otimes m}\}) \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes t}$, and the purity-testing property of the code $\{V_{k_1}\}_{k_1\in\mathcal{K}_1}$.

This concludes the proof that in the accept case, the real and the ideal scenarios are $\varepsilon/2$-close (i.e., the first term of Equation (3.32) is upper bounded by $\varepsilon/2$). The reject case is completely analogous.

Summing the accept and reject case as in Equation (3.32), we see that

$$\frac{1}{2}\left\|\mathfrak{I}(\rho)-\mathfrak{R}(\rho)\right\|_{\mathrm{tr}} \leqslant \varepsilon \tag{3.46}$$

for all ρ. This concludes the proof. □

3.6 Strong purity testing implies QCA-R

We are now ready for one of the main results of this chapter: the fact that Construction 3.2.8, when used on a strong-purity-testing code, results in a quantum ciphertext authentication code with key recycling. In the accept case, all key can be recycled; in the reject case, the one-time-pad key needs to be refreshed.

Theorem 3.6.1. *Let $\{V_{k_1}\}_{k_1 \in \mathcal{K}_1}$ be a strong-purity-testing code with error ε. The encryption scheme resulting from Construction 3.2.8 is quantum ciphertext $(\sqrt{\varepsilon} + \frac{3}{2}\varepsilon)$-authenticating with key recycling* (QCA-R)*, with recycling function $f(k_1, k_2) := k_1$.*

The proof outline is similar to the proof of Theorem 3.5.1, except that (part of) the secret key is appended to the state in the real and ideal scenarios. This extra register containing the secret key prevents us from immediately applying the Pauli twirl, as in the proof of Theorem 3.5.1. We thus have to employ some heavier machinery, again inspired by Portmann's proof [Por17, Theorem 3.5]. To preserve the flow of the proof, we postpone some derivation details to Section 3.6.1.

Proof. Let $\mathcal{A}$ be an adversary as in Definition 3.2.4. Define a simulator $\mathcal{S}$ in the same way as in the proof of Theorem 3.5.1. Note that this simulator is of the required form in the accept case (see Definition 3.2.4).

We show that for this simulator, the distance $\frac{1}{2}\|\mathfrak{I} - \mathfrak{R}\|_\diamond$ between the ideal and the real channel is upper bounded by $\sqrt{\varepsilon} + \frac{3}{2}\varepsilon$. Let ρ^{MRE} be any quantum state on the message register, side-information register, and an environment register E. Assume, as in the proof of Theorem 3.5.1, that $\mathcal{A}$ is a unitary map U^{MTR}. Let $\mu^{\mathsf{real}}_{k_1,k_2}$ and $\mu^{\mathsf{ideal}}_{k_1,k_2}$ be the effective output states in the real and ideal world, respectively:

$$\mu^{\mathsf{real}}_{k_1,k_2} := V^\dagger_{k_1} P^\dagger_{k_2} U^{MTR} P^{MT}_{k_2} V^{MT}_{k_1} \left(\rho \otimes |0^t\rangle\langle 0^t|\right) V^\dagger_{k_1} P^\dagger_{k_2} U^\dagger P_{k_2} V_{k_1}, \tag{3.47}$$

$$\mu^{\mathsf{ideal}}_{k_1,k_2} := V^\dagger_{k_1} P^\dagger_{k_2} U^{M_1TR} P^{M_1T}_{k_2} V^{M_1T}_{k_1} \left(\rho \otimes |0^t,\Phi^+\rangle\langle 0^t,\Phi^+|\right) V^\dagger_{k_1} P^\dagger_{k_2} U^\dagger P_{k_2} V_{k_1}. \tag{3.48}$$

Then we can write the result of the real and the ideal channels as

$$\begin{aligned}\mathfrak{R}(\rho) = \mathop{\mathbb{E}}_{k_1,k_2} \Bigg[& \langle 0^t|^T \mu^{\mathsf{real}}_{k_1,k_2} |0^t\rangle \otimes |k_1k_2\rangle\langle k_1k_2| \\ & + |\bot\rangle\langle\bot|^M \otimes \mathrm{Tr}_M\Bigg(\sum_{i\neq 0^t} \langle i|^T \mu^{\mathsf{real}}_{k_1,k_2} |i\rangle\Bigg) \otimes |k_1\rangle\langle k_1|\Bigg],\end{aligned} \tag{3.49}$$

$$\begin{aligned}\mathfrak{I}(\rho) = \mathop{\mathbb{E}}_{k_1',k_2'} \Bigg[& \langle \Phi^+,0^t|^{M_1M_2T} \mu^{\mathsf{ideal}}_{k_1',k_2'} |\Phi^+0^t\rangle \otimes \tau_{\mathcal{K}} \\ & + |\bot\rangle\langle\bot|^M \otimes \mathrm{Tr}_M\Bigg(\sum_{i\neq(\Phi^+,0^t)} \langle i|^{M_1M_2T} \mu^{\mathsf{ideal}}_{k_1',k_2'} |i\rangle\Bigg) \otimes \tau_{\mathcal{K}_1}\Bigg].\end{aligned} \tag{3.50}$$

These expressions are obtained simply by plugging in the description of the authentication scheme (see Construction 3.2.8) and the simulator into the channels of Definition 3.2.4. Since the accept states are orthogonal to the reject states in the M register, and since the key states are all mutually orthogonal, the distance $\frac{1}{2}\left\|\mathfrak{I}(\rho) - \mathfrak{R}(\rho)\right\|_{\mathrm{tr}}$

can be written as

$$\mathop{\mathbb{E}}_{k_1,k_2}\frac{1}{2}\left\|\mathop{\mathbb{E}}_{k_1',k_2'}\left(\langle\Phi^+,0^t|\,\mu^{\mathsf{ideal}}_{k_1',k_2'}\,|\Phi^+,0^t\rangle\right)-\langle 0^t|\,\mu^{\mathsf{real}}_{k_1,k_2}\,|0^t\rangle\right\|_{\mathrm{tr}}$$
$$+\ \mathop{\mathbb{E}}_{k_1}\frac{1}{2}\left\|\mathop{\mathbb{E}}_{k_1',k_2'}\left(\mathrm{Tr}_M\sum_{i\neq(0^t,\Phi^+)}\langle i|\,\mu^{\mathsf{ideal}}_{k_1',k_2'}\,|i\rangle\right)-\mathop{\mathbb{E}}_{k_2}\left(\mathrm{Tr}_M\sum_{i\neq 0^t}\langle i|\,\mu^{\mathsf{real}}_{k_1,k_2}\,|i\rangle\right)\right\|_{\mathrm{tr}}. \tag{3.51}$$

For a full derivation, see Section 3.6.1. We can thus focus on bounding the two terms in Equation (3.51), for accept and reject, separately. Note the difference between the two terms: in the reject case, the expectation over the one-time pad key k_2 does not have to be brought outside of the trace norm, since it is not recycled after a reject. This will make bounding the second term in Equation (3.51) the simpler of the two, so we will start with that one.

Decompose the attack as $U^{MTR}=\sum_\ell \alpha_\ell P_\ell^{MT}\otimes U_\ell^R$. Intuitively, the two states inside the second trace norm differ on those Paulis P_ℓ that are rejected in the ideal scenario, but not in the real one. The strong-purity-testing property promises that these Paulis are very few. However, we have to be careful, because the simulator independently generates its own set of keys. We will now bound the second term in Equation (3.51) more formally.

By rearranging sums, commuting Paulis, and applying projectors (for details: see Section 3.6.1), we can rewrite the second term inside the trace norm, the state in the real reject case for k_1, as

$$\mathop{\mathbb{E}}_{k_2}\left(\mathrm{Tr}_M\sum_{i\neq 0^t}\langle i|\,\mu^{\mathsf{real}}_{k_1,k_2}\,|i\rangle\right)=\mathrm{Tr}_M\left(\sum_{\ell\,:\,V_{k_1}^\dagger P_\ell V_{k_1}\notin\mathscr{P}_{\mathsf{real}}}|\alpha_\ell|^2\,U_\ell^R\rho^{MR}U_\ell^\dagger\right), \tag{3.52}$$

where $\mathscr{P}_{\mathsf{real}}$ contains the Paulis that are accepted by the real projector, i.e., $\mathscr{P}_{\mathsf{real}}:=\mathscr{P}_m\otimes\{\mathsf{I},\mathsf{Z}\}^{\otimes t}$. Similarly, defining $\mathscr{P}_{\mathsf{ideal}}:=\{\mathsf{I}^{\otimes m}\}\otimes\{\mathsf{I},\mathsf{Z}\}^{\otimes t}$ to be the set of Paulis that are allowed by the ideal projector, the resulting state in the reject case is

$$\mathop{\mathbb{E}}_{k_1',k_2'}\left(\mathrm{Tr}_M\sum_{i\neq(0^t,\Phi^+)}\langle i|\,\mu^{\mathsf{ideal}}_{k_1',k_2'}\,|i\rangle\right)=\mathrm{Tr}_M\left(\sum_{\ell\neq 0}\ \mathop{\mathbb{E}}_{\substack{k_1'\in\mathcal{K}_1\\ V_{k_1'}^\dagger P_\ell V_{k_1'}\notin\mathscr{P}_{\mathsf{ideal}}}}|\alpha_\ell|^2\,U_\ell^R\rho^{MR}U_\ell^\dagger\right) \tag{3.53}$$

$$\approx_\varepsilon \mathrm{Tr}_M\left(\sum_{\ell\neq 0}\mathop{\mathbb{E}}_{k_1'\in\mathcal{K}_1}|\alpha_\ell|^2\,U_\ell^R\rho^{MR}U_\ell^\dagger\right). \tag{3.54}$$

The approximation follows from the strong-purity-testing property of the code: the two states differ in those keys k_1' for which $V_{k_1'}^\dagger P_\ell V_{k_1'}\in\mathscr{P}_{\mathsf{ideal}}\subseteq\mathscr{P}_{\mathsf{real}}$, and for any nonidentity Pauli P_ℓ, this set is small by strong purity testing. Combined with the facts that $\mathrm{Tr}(U_\ell \rho U_\ell^\dagger) = 1$ and $\sum_\ell |\alpha_\ell|^2 = 1$, it follows that the states in Equations (3.53) and (3.54) are ε-close. Note that none of the terms in Equation (3.54) depend on k_1', so we can remove the expectation over it.

Applying the triangle inequality (twice), the second term in Equation (3.51) is found to be small:

$$\mathop{\mathbb{E}}_{k_1} \frac{1}{2} \left\| \mathop{\mathbb{E}}_{k_1',k_2'} \left(\mathrm{Tr}_M \sum_{i \neq (0^t, \Phi^+)} \langle i | \mu^{\text{ideal}}_{k_1',k_2'} | i \rangle \right) - \mathop{\mathbb{E}}_{k_2} \left(\mathrm{Tr}_M \sum_{i \neq 0^t} \langle i | \mu^{\text{real}}_{k_1,k_2} | i \rangle \right) \right\|_{\mathrm{tr}} \tag{3.55}$$

$$\leqslant \frac{\varepsilon}{2} + \mathop{\mathbb{E}}_{k_1} \frac{1}{2} \left\| \mathrm{Tr}_M \left(\sum_{\ell \,:\, V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{real}} \setminus \{ \mathsf{I}^{\otimes(m+t)} \}} |\alpha_\ell|^2 \, U_\ell \rho U_\ell^\dagger \right) \right\|_{\mathrm{tr}} \tag{3.56}$$

$$\leqslant \frac{\varepsilon}{2} + \frac{1}{2} \mathop{\mathbb{E}}_{k_1} \sum_{\ell \,:\, V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{real}} \setminus \{ \mathsf{I}^{\otimes(m+t)} \}} |\alpha_\ell|^2 , \tag{3.57}$$

which we can upper bound by ε by applying the strong-purity-testing property once more.

Next, we bound the first term of Equation (3.51): the difference between the ideal and the real channel in the accept case. The strategy is identical to the reject case that we just treated, but because we want to recycle both k_1 and k_2 in the accept case, we have to be more careful. The state in the real scenario, $\langle 0^t | \mu^{\text{real}}_{k_1,k_2} | 0^t \rangle$, cannot be rewritten into the compact form of, e.g., Equation (3.52), because we cannot average over the Pauli key k_2. Using a technical lemma due to Portmann [Por17] and Jensen's inequality in order to take the expectation over the keys inside, we obtain the bound

$$\mathop{\mathbb{E}}_{k_1,k_2} \left\| \mathop{\mathbb{E}}_{k_1',k_2'} \left(\langle \Phi^+, 0^t | \mu^{\text{ideal}}_{k_1',k_2'} | \Phi^+, 0^t \rangle \right) - \langle 0^t | \mu^{\text{real}}_{k_1,k_2} | 0^t \rangle \right\|_{\mathrm{tr}} \leqslant \frac{\varepsilon}{2} + \sqrt{\varepsilon}. \tag{3.58}$$

For a full derivation, see Section 3.6.1.

We have now upper bounded $\frac{1}{2} \| \mathfrak{I}(\rho) - \mathfrak{R}(\rho) \|_{\mathrm{tr}} \leqslant \sqrt{\varepsilon} + \frac{3}{2}\varepsilon$ for any state ρ^{MRE}, resulting in $\frac{1}{2} \| \mathfrak{I} - \mathfrak{R} \|_\diamond \leqslant \sqrt{\varepsilon} + \frac{3}{2}\varepsilon$, as desired. □

3.6.1 Details for the proof of Theorem 3.6.1

Derivation of Equation (3.51)

We give details on how to arrive at Equation (3.51) given the expressions for $\mathfrak{I}$ and $\mathfrak{R}$ in the proof of Theorem 3.6.1.

$$\frac{1}{2} \| \mathfrak{I}(\rho) - \mathfrak{R}(\rho) \|_{\mathrm{tr}} \tag{3.59}$$

$$
\begin{aligned}
&= \frac{1}{2}\Bigg\| \mathop{\mathbb{E}}_{k_1',k_2'} \left[\langle \Phi^+, 0^t | \mu^{\mathsf{ideal}}_{k_1',k_2'} | \Phi^+, 0^t \rangle \otimes \tau_{\mathcal{K}} \right] \\
&\qquad + \mathop{\mathbb{E}}_{k_1',k_2'} \left[|\perp\rangle\langle\perp| \otimes \mathrm{Tr}_M \left(\sum_{i \neq (\Phi^+,0^t)} \langle i | \mu^{\mathsf{ideal}}_{k_1',k_2'} | i \rangle \right) \otimes \tau_{\mathcal{K}_1} \right] \\
&\quad - \mathop{\mathbb{E}}_{k_1,k_2} \left[\langle 0^t | \mu^{\mathsf{real}}_{k_1,k_2} | 0^t \rangle \otimes |k_1 k_2\rangle\langle k_1 k_2| \right] \\
&\qquad - \mathop{\mathbb{E}}_{k_1,k_2} \left[|\perp\rangle\langle\perp| \otimes \mathrm{Tr}_M \left(\sum_{i \neq 0^t} \langle i | \mu^{\mathsf{real}}_{k_1,k_2} | i \rangle \right) \otimes |k_1\rangle\langle k_1| \right] \Bigg\|_{\mathrm{tr}} \qquad (3.60) \\
&= \frac{1}{2}\Bigg\| \mathop{\mathbb{E}}_{k_1',k_2'} \left[\langle \Phi^+, 0^t | \mu^{\mathsf{ideal}}_{k_1',k_2'} | \Phi^+, 0^t \rangle \otimes \tau_{\mathcal{K}} \right] \\
&\qquad - \mathop{\mathbb{E}}_{k_1,k_2} \left[\langle 0^t | \mu^{\mathsf{real}}_{k_1,k_2} | 0^t \rangle \otimes |k_1 k_2\rangle\langle k_1 k_2| \right] \Bigg\|_{\mathrm{tr}} \\
&\quad + \frac{1}{2}\Bigg\| \mathop{\mathbb{E}}_{k_1',k_2'} \left[\mathrm{Tr}_M \left(\sum_{i \neq (\Phi^+,0^t)} \langle i | \mu^{\mathsf{ideal}}_{k_1',k_2'} | i \rangle \right) \otimes \tau_{\mathcal{K}_1} \right] \\
&\qquad - \mathop{\mathbb{E}}_{k_1,k_2} \left[\mathrm{Tr}_M \left(\sum_{i \neq 0^t} \langle i | \mu^{\mathsf{real}}_{k_1,k_2} | i \rangle \right) \otimes |k_1\rangle\langle k_1| \right] \Bigg\|_{\mathrm{tr}}, \qquad (3.61)
\end{aligned}
$$

because $\| \rho + \sigma \|_{\mathrm{tr}} = \| \rho \|_{\mathrm{tr}} + \| \sigma \|_{\mathrm{tr}}$ whenever ρ and σ are orthogonal (and the accept and reject states are orthogonal in the M register), and because $\| |a\rangle\langle a| \otimes \rho \|_{\mathrm{tr}} = \| \rho \|_{\mathrm{tr}}$ for basis states $|a\rangle$ (this allows us to get rid of the $|\perp\rangle\langle\perp|$).

Observing that the $|k_1 k_2\rangle\langle k_1 k_2|$ states (or $|k_1\rangle\langle k_1|$ in the case of reject) are all orthogonal to each other, we obtain Equation (3.51).

Derivation of Equation (3.52)

We give details on how Equation (3.52) is derived in the proof of Theorem 3.6.1.

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{k_2} \left(\mathrm{Tr}_M \sum_{i \neq 0^t} \langle i | \mu^{\mathsf{real}}_{k_1,k_2} | i \rangle \right) \qquad (3.62) \\
&= \mathop{\mathbb{E}}_{k_2} \Bigg(\mathrm{Tr}_M \sum_{i \neq 0^t} \langle i | V^\dagger_{k_1} P^\dagger_{k_2} \left(\sum_\ell \alpha_\ell P^{MT}_\ell \otimes U^R_\ell \right) P^{MT}_{k_2} V^{MT}_{k_1} \left(\rho^{MRE} \otimes |0^t\rangle\langle 0^t|^T \right) \\
&\qquad V^\dagger_{k_1} P^\dagger_{k_2} \left(\sum_{\ell'} \alpha^*_{\ell'} P_{\ell'} \otimes U^\dagger_{\ell'} \right) P_{k_2} V_{k_1} | i \rangle \Bigg) \qquad (3.63) \\
&= \mathop{\mathbb{E}}_{k_2} \Bigg(\mathrm{Tr}_M \sum_{i \neq 0^t} \sum_{\ell,\ell'} \alpha_\ell \alpha^*_{\ell'} \langle i | \left(V^\dagger_{k_1} P^\dagger_{k_2} P^{MT}_\ell P^{MT}_{k_2} V^{MT}_{k_1} \otimes U^R_\ell \right) \left(\rho^{MRE} \otimes |0^t\rangle\langle 0^t|^T \right) \\
&\qquad \left(V^\dagger_{k_1} P^\dagger_{k_2} P_{\ell'} P_{k_2} V_{k_1} \otimes U^\dagger_{\ell'} \right) | i \rangle \Bigg), \qquad (3.64)
\end{aligned}
$$

where the second equality is obtained by moving summation signs further out and rearranging the unitaries so that they are sorted according to the register they act on. Recall that the ℓ and ℓ' index an $(m+t)$-qubit Pauli, and thus consist of $2(m+t)$ bits: we can regard any such Pauli-index a as (x_a, z_a) where the two parts are $(m+t)$-bit strings describing the locations of the X and Z Paulis, respectively. Recall from Lemma 2.4.7 that the symplectic inner product $(a,b)_{\mathsf{Sp}} := x_a \cdot z_b - x_b \cdot z_a$ equals 1 whenever the Paulis P_a and P_b commute, and -1 when they anti-commute. We can then continue the derivation as follows:

$$= \mathop{\mathbb{E}}_{k_2} \left(\mathrm{Tr}_M \sum_{i \neq 0^t} \sum_{\ell,\ell'} \alpha_\ell \alpha^*_{\ell'} (-1)^{(\ell \oplus \ell', k_2)_{\mathsf{Sp}}} \langle i | \left(V^\dagger_{k_1} P_\ell V^{MT}_{k_1} \otimes U^R_\ell \right) \left(\rho^{MRE} \otimes |0^t\rangle\langle 0^t|^T \right) \right.$$
$$\left. \left(V^\dagger_{k_1} P_{\ell'} V_{k_1} \otimes U^\dagger_{\ell'} \right) |i\rangle \right) \tag{3.65}$$
$$= \mathrm{Tr}_M \sum_{i \neq 0^t} \sum_{\ell} |\alpha_\ell|^2 \langle i | \left(V^\dagger_{k_1} P_\ell V^{MT}_{k_1} \otimes U^R_\ell \right) \left(\rho \otimes |0^t\rangle\langle 0^t| \right) \left(V^\dagger_{k_1} P_\ell V_{k_1} \otimes U^\dagger_\ell \right) |i\rangle . \tag{3.66}$$

If we now apply the projector on the tag register T, and observe that the projection preserves exactly those ℓ such that $V^\dagger_{k_1} P_\ell V_{k_1} \notin \mathscr{P}_{\mathsf{real}}$ (i.e., those P_ℓ that are rejected), the right-hand side of Equation (3.52) is obtained.

Derivation of Equation (3.58)

We give details on how to obtain Equation (3.58), which bounds the difference between the accept and reject case in the real scenario.

For this derivation, it will be useful to purify ρ^{MRE} as $\sum_i \beta_i |\psi_i\rangle^{MREE'}$, where E' is an extra environment register for the purification. Note that $\langle \psi_i | \psi_j \rangle = 0$ for $i \neq j$.

With the same strategy as in the derivation of Equation (3.52), except for the last step, we can rewrite the real state as

$$\langle 0^t | \mu^{\mathsf{real}}_{k_1,k_2} |0^t\rangle \tag{3.67}$$
$$= \langle 0^t | V^\dagger_{k_1} P^\dagger_{k_2} \left(\sum_\ell \alpha_\ell P^{MT}_\ell \otimes U^R_\ell \right) P^{MT}_{k_2} V^{MT}_{k_1} \left(\sum_{i,i'} \beta_i \beta^*_{i'} |\psi_i\rangle\langle\psi_{i'}| \otimes |0^t\rangle\langle 0^t|^T \right)$$
$$V^\dagger_{k_1} P^\dagger_{k_2} \left(\sum_{\ell'} \alpha^*_{\ell'} P_{\ell'} \otimes U^\dagger_{\ell'} \right) P_{k_2} V_{k_1} |0^t\rangle \tag{3.68}$$
$$= \sum_{i,i'} \sum_{\ell,\ell'} \beta_i \alpha_\ell \beta^*_{i'} \alpha^*_{\ell'} \langle 0^t | \left(V^\dagger_{k_1} P^\dagger_{k_2} P_\ell P_{k_2} V^{MT}_{k_1} \otimes U^R_\ell \right) |\psi_i\rangle |0^t\rangle$$
$$\langle \psi_{i'} | \langle 0^t | \left(V^\dagger_{k_1} P^\dagger_{k_2} P_{\ell'} P_{k_2} V_{k_1} \otimes U^\dagger_{\ell'} \right) |0^t\rangle \tag{3.69}$$

$$= \sum_{i,i'} \sum_{\ell,\ell'} \beta_i \alpha_\ell \beta_{i'}^* \alpha_{\ell'}^* (-1)^{(\ell,k_2)_{\mathsf{Sp}}} (-1)^{(\ell',k_2)_{\mathsf{Sp}}} \langle 0^t | \left(V_{k_1}^\dagger P_\ell V_{k_1}^{MT} \otimes U_\ell^R \right) |\psi_i\rangle |0^t\rangle$$
$$\langle \psi_{i'} | \langle 0^t | \left(V_{k_1}^\dagger P_{\ell'} V_{k_1} \otimes U_{\ell'}^\dagger \right) |0^t\rangle, \tag{3.70}$$

which, as a pure state, can be expressed as

$$\sum_i \sum_\ell \beta_i \alpha_\ell (-1)^{(\ell,k_2)_{\mathsf{Sp}}} \langle 0^t | \left(V_{k_1}^\dagger P_\ell V_{k_1}^{MT} \otimes U_\ell^R \right) |\psi_i\rangle |0^t\rangle \tag{3.71}$$
$$= \sum_{\ell : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell (-1)^{(\ell,k_2)_{\mathsf{Sp}}} (Q_{k_1,\ell}^M \otimes U_\ell^R) \sum_i \beta_i |\psi_i\rangle, \tag{3.72}$$

where $Q_{k_1,\ell}$ is the Pauli on the message register M that results from $V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}} = \mathscr{P}_m \otimes \{\mathsf{I}, \mathsf{Z}\}^{\otimes t}$. In much the same way, but using Lemma 2.4.7 as we did in the derivation of Equation (3.52), we can write the accept state in the ideal scenario,

$$\mathop{\mathbb{E}}_{k_1',k_2'} \left(\langle \Phi^+, 0^t | \mu_{k_1',k_2'}^{\mathsf{ideal}} |\Phi^+, 0^t\rangle \right), \tag{3.73}$$

as the pure state

$$\mathop{\mathbb{E}}_{k_1'} \left(\sum_{\ell : V_{k_1'}^\dagger P_\ell V_{k_1'} \in \mathscr{P}_{\mathsf{ideal}}} \alpha_\ell U_\ell^R \sum_i \beta_i |\psi_i\rangle \right) \tag{3.74}$$
$$= \sum_\ell \mathop{\mathbb{E}}_{k_1'} \delta_{(V_{k_1'}^\dagger P_\ell V_{k_1'} \in \mathscr{P}_{\mathsf{ideal}})} \alpha_\ell U_\ell^R \sum_i \beta_i |\psi_i\rangle, \tag{3.75}$$

where $\delta_{(V_{k_1'}^\dagger P_\ell V_{k_1'} \in \mathscr{P}_{\mathsf{ideal}})}$ is the indicator function that is equal to 1 whenever $V_{k_1'}^\dagger P_\ell V_{k_1'} \in \mathscr{P}_{\mathsf{ideal}}$, and 0 otherwise. Continuing, we rewrite the ideal accept state as

$$= \alpha_0 U_0^R \sum_i \beta_i |\psi_i\rangle \;+\; \sum_{\ell \neq 0} \mathop{\mathbb{E}}_{k_1'} \delta_{(V_{k_1'}^\dagger P_\ell V_{k_1'} \in \mathscr{P}_{\mathsf{ideal}})} \alpha_\ell U_\ell^R \sum_i \beta_i |\psi_i\rangle. \tag{3.76}$$

(Recall that $P_0 = \mathsf{I}^{\otimes(m+t)}$ by convention.) By the strong-purity-testing property of the code $\{V_{k_1}\}_{k_1 \in \mathcal{K}_1}$, and the fact that $\mathscr{P}_{\mathsf{ideal}} \subseteq \mathscr{P}_{\mathsf{real}}$, the second term in Equation (3.76) has very small amplitude. Thus, the ideal accept state is ε-close in trace distance to $\alpha_0 U_0^R \sum_i \beta_i |\psi_i\rangle$.

We are now ready to bound the expected distance between the ideal and the real case:

$$\mathop{\mathbb{E}}_{k_1,k_2} \frac{1}{2} \left\| \mathop{\mathbb{E}}_{k_1',k_2'} \left(\langle \Phi^+, 0^t | \mu_{k_1',k_2'}^{\mathsf{ideal}} |\Phi^+, 0^t\rangle \right) - \langle 0^t | \mu_{k_1,k_2}^{\mathsf{real}} |0^t\rangle \right\|_{\mathrm{tr}} \tag{3.77}$$

$$
\begin{aligned}
&\leqslant \frac{\varepsilon}{2} + \mathop{\mathbb{E}}_{k_1,k_2} \frac{1}{2} \Bigg\| \, |\alpha_0|^2 \sum_{i,i'} \beta_i \beta_{i'} U_0 \left|\psi_i\right\rangle\left\langle\psi_{i'}\right| U_0^\dagger \\
&\quad - \left(\sum_{\ell : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell (-1)^{(\ell,k_2)_{\mathsf{Sp}}} (Q_{k_1,\ell} \otimes U_\ell) \sum_i \beta_i \left|\psi_i\right\rangle \right) \\
&\quad \cdot \left(\sum_{\ell' : V_{k_1}^\dagger P_{\ell'} V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_{\ell'} (-1)^{(\ell',k_2)_{\mathsf{Sp}}} \sum_i \beta_i \left\langle\psi_i\right| (Q_{k_1,\ell'} \otimes U_{\ell'}^\dagger) \right) \Bigg\|_{\mathrm{tr}} \qquad (3.78) \\
&\leqslant \frac{\varepsilon}{2} + \mathop{\mathbb{E}}_{k_1,k_2} \left\| \alpha_0 U_0 \sum_i \beta_i \left|\psi_i\right\rangle - \left(\sum_{\ell : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell (-1)^{(\ell,k_2)_{\mathsf{Sp}}} (Q_{k_1,\ell} \otimes U_\ell) \sum_i \beta_i \left|\psi_i\right\rangle \right) \right\|_2 . \qquad (3.79)
\end{aligned}
$$

The first inequality is the triangle inequality, and the second one follows from the fact that $\frac{1}{2} \| \left|\varphi\right\rangle\left\langle\varphi\right| - \left|\psi\right\rangle\left\langle\psi\right| \|_{\mathrm{tr}} \leqslant \| \left|\varphi\right\rangle - \left|\psi\right\rangle \|_2$ [Por17, Lemma C.2] (where the latter is the vector 2-norm, see Section 2.4.3).

Abbreviate $\sigma_{i,i',\ell,\ell',k_1,k_2} := (-1)^{(\ell \oplus \ell', k_2)_{\mathsf{Sp}}} \left\langle\psi_i\right| (Q_{k_1,\ell} \otimes U_\ell^\dagger)(Q_{k_1,\ell'} \otimes U_\ell) \left|\psi_{i'}\right\rangle$. Then, continuing our derivation from Equation (3.79),

$$
\begin{aligned}
&= \frac{\varepsilon}{2} + \mathop{\mathbb{E}}_{k_1,k_2} \left\| \sum_i \beta_i \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell (-1)^{(\ell,k_2)_{\mathsf{Sp}}} (Q_{k_1,\ell} \otimes U_\ell) \left|\psi_i\right\rangle \right\|_2 \qquad (3.80) \\
&= \frac{\varepsilon}{2} + \mathop{\mathbb{E}}_{k_1,k_2} \sqrt{ \sum_{i,i'} \beta_i^* \beta_{i'} \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \sum_{\ell' \neq 0 : V_{k_1}^\dagger P_{\ell'} V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell^* \alpha_{\ell'} \sigma_{i,i',\ell,\ell',k_1,k_2} } \qquad (3.81) \\
&\leqslant \frac{\varepsilon}{2} + \sqrt{ \mathop{\mathbb{E}}_{k_1,k_2} \sum_{i,i'} \beta_i^* \beta_{i'} \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \sum_{\ell' \neq 0 : V_{k_1}^\dagger P_{\ell'} V_{k_1} \in \mathscr{P}_{\mathsf{real}}} \alpha_\ell^* \alpha_{\ell'} \sigma_{i,i',\ell,\ell',k_1,k_2} } \qquad (3.82) \\
&= \frac{\varepsilon}{2} + \sqrt{ \mathop{\mathbb{E}}_{k_1} \sum_{i,i'} \beta_i^* \beta_{i'} \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} |\alpha_\ell|^2 \left\langle\psi_i\right| (Q_{k_1,\ell} \otimes U_\ell^\dagger)(Q_{k_1,\ell} \otimes U_\ell) \left|\psi_{i'}\right\rangle }, \qquad (3.83)
\end{aligned}
$$

by Jensen's inequality and by Lemma 2.4.7. This last line can be greatly simplified to

$$
\begin{aligned}
&= \frac{\varepsilon}{2} + \sqrt{ \mathop{\mathbb{E}}_{k_1} \sum_i |\beta_i|^2 \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} |\alpha_\ell|^2 \left\langle\psi_i\middle|\psi_i\right\rangle } \qquad (3.84) \\
&= \frac{\varepsilon}{2} + \sqrt{ \mathop{\mathbb{E}}_{k_1} \sum_{\ell \neq 0 : V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\mathsf{real}}} |\alpha_\ell|^2 } \qquad (3.85)
\end{aligned}
$$

$$= \frac{\varepsilon}{2} + \sqrt{\sum_{\ell \neq 0} \mathop{\mathbb{E}}_{k_1} \delta_{(V_{k_1}^\dagger P_\ell V_{k_1} \in \mathscr{P}_{\text{real}})} |\alpha_\ell|^2} \tag{3.86}$$

$$\leqslant \frac{\varepsilon}{2} + \sqrt{\sum_{\ell \neq 0} \varepsilon |\alpha_\ell|^2} \tag{3.87}$$

$$\leqslant \frac{\varepsilon}{2} + \sqrt{\varepsilon}. \tag{3.88}$$

The first inequality is again due to the strong-purity-testing property of the code $\{V_{k_1}\}_{\mathcal{K}_1}$. This concludes the derivation of Equation (3.58).

3.7 The strong trap code is strong purity testing

Theorem 3.6.1 already gives us a quantum-ciphertext-authenticating code with key recycling: the Clifford code. However, as explained in Section 3.3, the Clifford code requires significant resources from the encoding party, making it less suitable for delegated quantum computation. In this section, we therefore present a strong-purity-testing variation on the trap code, the *strong trap code* (already briefly discussed in Section 3.3.4), which does allow for computation on the ciphertexts in a meaningful and efficient way. By Theorem 3.6.1, this construction immediately gives rise to a ciphertext authentication scheme with key recycling (QCA-R).

The strong trap code will be based on a family of CSS codes. In Sections 3.7.1 and 3.7.2, we describe the properties that we want this family of codes to satisfy. In Section 3.7.3, we construct a family of CSS codes that satisfies those two properties, and in Section 3.7.4, we will use this family to detail the construction of the strong trap code and to prove that it is strong purity testing.

The strong trap code requires the existence of a family of quantum error-correcting codes with two specific properties: a high benign distance, and weight sparsity. We define these properties first.

3.7.1 Benign distance

If a QECC has distance d, it is not necessarily able to detect all Pauli errors of weight less than d. For example, if one of the qubits in a codeword is in the state $|0\rangle$, then a Pauli-Z remains undetected. In general, any Pauli error that stabilizes all codewords will remain undetected by the code. Of course, such an error does not directly cause harm or adds noise to the state, because it effectively performs the identity operation. However, in an adversarial setting, even such "benign" Pauli errors indicate that someone tried to modify the state.

As an alternative to Definition 3.2.5, we consider a distance measure for quantum error-correcting codes that describes the lowest possible weight of a stabilizer:

Definition 3.7.1 (Benign distance)**.** The *benign distance* of an $[[n,m]]$ code is the minimum weight of a nonidentity Pauli P_ℓ such that $P_\ell |x_L\rangle = |x_L\rangle$ for all $x \in \{0,1\}^m$. If such P_ℓ does not exist, the benign distance is ∞.

To distinguish the benign distance from the notion of distance defined in Definition 3.2.5, we will often use the term *conventional distance* to refer to the latter.

The benign distance is not in a fixed relation to the conventional distance. For example, the $[[7,4]]$ Steane code has distance 3, but benign distance 4. On the other hand, the $[[49,1]]$ concatenated Steane code has distance 9, but a benign distance of only 4 (any nonidentity stabilizer for the $[[7,4]]$ Steane code is also a stabilizer on the $[[49,1]]$ code if it is concatenated with identity on the other blocks). Even though the two quantities do not bound each other in general, we observe that the benign distance of weakly self-dual CSS codes (i.e., CSS codes constructed from a weakly self-dual classical code) grows with their conventional distance:

Lemma 3.7.2. *Let C_1 an C_2 be two classical linear codes such that $C_2 \subseteq C_1$ and $C_1^\perp = C_2$. Then the benign distance of $CSS(C_1, C_2)$ is greater than or equal to its conventional distance.*

Proof. Let d denote the distance of the classical code C_1. By the construction of CSS codes, $CSS(C_1, C_2)$ also has (conventional) distance d.

Also by construction, the check matrix of $CSS(C_1, C_2)$ is given by

$$\left[\begin{array}{c|c} H(C_2^\perp) & 0 \\ 0 & H(C_1) \end{array}\right] = \left[\begin{array}{c|c} G(C_2) & 0 \\ 0 & G(C_2) \end{array}\right], \tag{3.89}$$

where $G(\cdot)$ the generator matrix (see Equation (3.3)).

The rows of $G(C_2)$ form a basis for the codewords in C_2. Since C_1 has distance d, and $C_2 \subseteq C_1$, any row in $G(C_2)$, and any linear combination of these rows, has weight at least d. Thus, any linear combination of rows in the above check matrix also has weight at least d.

The rows of the check matrix generate the stabilizers of the code (see Section 3.2.2). We conclude that the stabilizers of the code $CSS(C_1, C_2)$ all have weight at least d, and therefore the benign distance of the code is at least d. □

Furthermore, if the CSS code is built from a so-called *punctured* classical code, then its distance (and therefore benign distance) is high.

Lemma 3.7.3. *Let C be a classical $[n,m,d]$ self-dual linear code for some $d > 1$. Assume w.l.o.g.*[6] *that not all codewords in C end in 0. Define*

$$C_1 := \{c \in \{0,1\}^{n-1} \mid c0 \in C \vee c1 \in C\}$$

and $C_2 := C_1^\perp$. Then $CSS(C_1, C_2)$ is an $[[n-1, 1, d']]$ code with $d' \in \{d-1, d\}$ and with benign distance $d_b \geq d' \geq d-1$.

[6] If they do, pick a different position to puncture at. Since $d \neq 0$, such a position always exists.

Proof. The code C_1 is a $[n-1, m, d']$ code for some $d' \in \{d-1, d\}$. Firstly, it has length $n-1$, because one bit is removed (punctured) from the codewords of C. Secondly, it has rank m: there are no two codewords in C that differ at only the punctured bit (since $d > 1$), and so the punctured versions of two distinct codewords are also distinct. Thirdly, it has distance $d-1$, again because one bit is removed from the codewords in C which all had weight at least d.

In order to prove the statement of the lemma, we need to show two things: firstly, that C_2 has rank $m-1$ (from which the parameters of the CSS code will follow), and secondly, that $C_2 \subseteq C_1$ (from which $d_b \geqslant d'$ will follow by Lemma 3.7.2, and which shows that C_1 and C_2 are valid candidates for the CSS construction).

We start by showing a stronger version of the second claim, namely that $C_2 = \{c_2 \mid c_2 0 \in C\}$. The latter set is then clearly a subset of C_1. For the forward inclusion, pick an arbitrary $c_2 \in C_2$. For all $c \in C$, which by definition of C_1 we can write as $c = c_1 b$ for $c_1 \in C_1$ and $b \in \{0,1\}$, it follows that $\langle c_2 0, c\rangle = \langle c_2 0, c_1 b\rangle = \langle c_2, c_1\rangle + 0 = 0$. The last equality follows from the fact that $C_2 = C_1^\perp$. And so, $c_2 0 \in C^\perp = C$ (as C is self-dual). For the other inclusion, pick a $c_2 \in \{c_2 \mid c_2 0 \in C\}$. To show that $c_2 \in C_2 = C_1^\perp$, we show that $\langle c_1, c_2\rangle = 0$ for all $c_1 \in C_1$: let $c_1 \in C_1$, and note that by definition, there is a $b \in \{0,1\}$ such that $c_1 b \in C$. Then $\langle c_1, c_2\rangle = \langle c_1 b, c_2 0\rangle = 0$, as C is self-dual.

It remains to show that $|C_2| = \frac{1}{2}|C_1|$, i.e., C_2 has rank $m-1$. For this, the stronger statement $C_2 = \{c \mid c0 \in C\}$ proven above will be useful. To see that $|\{c \mid c0 \in C\}| = \frac{1}{2}|C_1|$, consider a basis $\{v_1, \ldots, v_m\}$ for C, and let $I \subseteq [m]$ be the set of indices i such that $v_i[n] = 1$ (recall that we assumed without loss of generality that $I \neq \emptyset$). Note that all $x \in \{0,1\}^m$ represent a (unique) codeword $c = x_1 v_1 + x_2 v_2 + \cdots + x_m v_m$ of C, and conversely every codeword in C is represented by some $x \in \{0,1\}^m$ as $x_1 v_1 + x_2 v_2 + \cdots x_m v_m$. Since I is nonempty, exactly half of all $x \in \{0,1\}^m$ have $\sum_{i \in I} x_i = 0 \mod 2$ (resulting in the nth bit of c being 0), and exactly half have $\sum_{i \in I} x_i = 1 \mod 2$ (resulting in the nth bit of c being 1). Thus, exactly half of the elements in C are of the form $a0$ for some $a \in \{0,1\}^{n-1}$. Since $d > 1$, and so distinct codewords in C are punctured to distinct codewords in C_1, the statement $|\{c \mid c0 \in C\}| = \frac{1}{2}|C_1|$ follows.

We may conclude that the rank of C_2 is $m-1$. Thus, $CSS(C_1, C_2)$ is an $[[n-1, 1, d']]$ code for $d' \in \{d-1, d\}$. □

3.7.2 Weight sparsity

We define a second property of interest: *weight sparsity*. Intuitively, weight sparsity means that for any set of X-, Y-, and Z-weights, randomly selecting a Pauli operator with those weights only yields a stabilizer with very small probability. This probability should shrink whenever the codeword length grows; for this reason, we consider weight sparsity as a property of code *families* rather than of individual codes.

Definition 3.7.4 (Weight-sparse code family)**.** Let $(E_i)_{i \in \mathbb{N}}$ be a family of quantum error-correcting codes with parameters $[[n(i), m(i), d(i)]]$. For each $i \in \mathbb{N}$, and for all nonnegative integers x, y, z such that $x + y + z \leqslant n(i)$, let $A_i(x, y, z)$ denote the set of $n(i)$-qubit Paulis with X-weight x, Y-weight y, and Z-weight z. Let $B_i(x, y, z)$ denote the set of benign Paulis in $A_i(x, y, z)$.

The family $(E_i)_{i\in\mathbb{N}}$ is *weight-sparse* if the *maximum benign ratio*

$$f(i) := \max_{\substack{x,y,z\\ x+y+z\leqslant n(i)}} \frac{|B_i(x, y, z)|}{|A_i(x, y, z)|}$$

is negligiblein $n(i)$.

As it turns out, for CSS-code families it suffices to show that the family is X-weight sparse, as illustrated by the definition and lemma below.

Definition 3.7.5 (X-weight-sparse code family)**.** Let $(E_i)_{i\in\mathbb{N}}$ be a family of quantum error-correction codes with parameters $[[n(i), m(i), d(i)]]$, and define the sets $A_i(x, y, z)$ and $B_i(x, y, z)$ as in Definition 3.7.4. Moreover, define $A_i^{\mathsf{X}}(x) := A_i(x, 0, 0)$ and $B_i^{\mathsf{X}}(x) := B_i(x, 0, 0)$.

The family $(E_i)_{i\in\mathbb{N}}$ is X-*weight sparse* if the function

$$f_{\mathsf{X}}(i) := \max_{w\leqslant n(i)} \frac{|B_i^{\mathsf{X}}(w)|}{|A_i^{\mathsf{X}}(w)|}$$

is negligible in $n(i)$.

Note that for weakly self-dual CSS codes, X-weight sparsity immediately implies Z-weight sparsity (defined analogously): by construction, the stabilizers of a weakly self-dual CSS code are are symmetric in their X and Z stabilizers (see Equation (3.89)). It also implies general weight sparsity:

Lemma 3.7.6. *If the codes in the family $(E_i)_{i\in\mathbb{N}}$ are all CSS codes, and the family is* X*-weight sparse, then the family is also weight sparse.*

Proof. Recall from Section 3.2.2 that a CSS code has a stabilizer generating set containing elements that are built up of either exclusively X and I (we will call these X-stabilizers), or exclusively Z and I (which we will call Z-stabilizers). Thus, any stabilizer P_ℓ for the CSS code can be written as a product $P_{\ell_x} P_{\ell_z}$ of an X-stabilizer P_{ℓ_x} and a Z-stabilizer P_{ℓ_z}.

Consider a code E_i in the family, and arbitrary nonnegative integers x, y, z such that $x + y + z \leqslant n(i)$. Then every element of $A_i(x, y, z)$ can be constructed by first selecting a Pauli P_{ℓ_x} of the appropriate weight, and then selecting a Pauli P_{ℓ_z} of the appropriate weight that overlaps with P_{ℓ_x} at an appropriate number of positions:

$$A_i(x, y, z) = \{P_{\ell_x} P_{\ell_z} \mid P_{\ell_x} \in A_i^{\mathsf{X}}(x + y) \text{ and } P_{\ell_z} \in A_i^{\mathsf{Z}}(y + z \mid P_{\ell_x}, y)\}, \tag{3.90}$$

where $A_i^{\mathsf{Z}}(y+z \mid P_{\ell_x}, y)$ denotes the subset of $A_i(0,0,x+y)$ that overlap with P_{ℓ_x} on exactly y positions. Note that for all $P_{\ell_x} \in A_i^{\mathsf{X}}(x+y)$,

$$\left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_x}, y)\right| = \left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right|, \tag{3.91}$$

for the canonical $P_{\ell_0} := \mathsf{X}^{\otimes(x+y)}\mathsf{I}^{\otimes(n(i)-x-y)}$. Hence,

$$\left|A_i(x,y,z)\right| = \left|A_i^{\mathsf{X}}(x+y)\right| \cdot \left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right|. \tag{3.92}$$

Similarly, define $B_i^{\mathsf{Z}}(y+z \mid P_{\ell_x}, y)$ to be the benign subset of $A_i^{\mathsf{Z}}(y+z \mid P_{\ell_x}, y)$. Using the same reasoning as above, we can arrive at the inequality

$$\left|B_i(x,y,z)\right| = \left|B_i^{\mathsf{X}}(x+y)\right| \cdot \left|B_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right| \leqslant \left|B_i^{\mathsf{X}}(x+y)\right| \cdot \left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right|. \tag{3.93}$$

Combining the above results, we conclude that for all nonnegative integers x, y, and z such that $x+y+z \leqslant n(i)$,

$$\frac{\left|B_i(x,y,z)\right|}{\left|A_i(x,y,z)\right|} \leqslant \frac{\left|B_i^{\mathsf{X}}(x+y)\right| \cdot \left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right|}{\left|A_i^{\mathsf{X}}(x+y)\right| \cdot \left|A_i^{\mathsf{Z}}(y+z \mid P_{\ell_0}, y)\right|} = \frac{\left|B_i^{\mathsf{X}}(x+y)\right|}{\left|A_i^{\mathsf{X}}(x+y)\right|}. \tag{3.94}$$

Maximizing over x, y, z on both sides of the inequality, and noting that $x+y \leqslant n(i)$, the statement of the lemma follows. □

3.7.3 A high-benign-distance, weight-sparse QECC family

In this subsection, we construct a family of quantum error-correcting codes from a punctured version of classical Reed–Muller codes [Sho96; Pre97]. We show that it has distance and benign distance $O(\sqrt{n(i)})$, where $n(i)$ is the codeword length of the ith code in the family, and that it is weight sparse.

Reed–Muller codes are a class of codes based on polynomials on the field $\mathbb{F}_2^a$ for some $a \in \mathbb{N}$. Every polynomial on the field is associated with a vector of length 2^a, representing the values of that polynomial on every possible input. The Reed–Muller code $R(i,a)$ consists of all these vectors for polynomials of degree up to i. Increasing i while keeping a constant results in a higher-ranked code, but with a smaller distance. For a discussion, see [Pre97, Chapter 7].

For our purposes, we will be interested in Reed–Muller codes where a scales with i as $a = 2i+1$ (for $i \in \mathbb{N}$), as these codes happen to be self-dual. They have codeword length 2^{2i+1}, rank 2^{2i}, and distance 2^{i+1} [Pre97]. Instantiating the construction in Lemma 3.7.3 with $R(i, 2i+1)$, we see that the resulting quantum code R_i is an $[[n(i), m(i), d(i)]]$ code with $n(i) = 2^{2i+1} - 1$, $m(i) = 1$, and $d(i) = 2^{i+1} - 1$. In the resulting family $(R_i)_{i \in \mathbb{N}}$, the codeword length grows approximately quadratically with the desired distance (since $n(i) = \frac{1}{2}(d(i)+1)^2 + 1$). The benign distance is also high, at least $d(i) - 1$.

Lemma 3.7.7. *The family $(R_i)_{i\in\mathbb{N}}$ of quantum error-correction codes, where R_i is constructed by puncturing the self-dual Reed–Muller code $R(i, 2i+1)$, is weight sparse.*

Proof. Since all the codes R_i are CSS codes, by Lemma 3.7.6 we only need to show that the family is X-weight sparse, i.e.,

$$\max_{w\leqslant n(i)} \frac{\left|B_i^{\mathsf{X}}(w)\right|}{\left|A_i^{\mathsf{X}}(w)\right|} \leqslant \mathrm{negl}(n(i)) \tag{3.95}$$

Recall from Lemma 3.7.6 that $A_i^{\mathsf{X}}(w)$ refers to the set of Paulis consisting of w Pauli-Xs and $n(i)-w$ identity operations, in any order. B_i^{X} is the benign subset of A_i^{X}.

The cardinality of $A_i^{\mathsf{X}}(w)$ can be expressed as $\binom{n(i)}{w}$. Define $C_{i,1} := R_i$ and $C_{i,2} := R_i^{\perp}$. By construction of R_i, all its X-stabilizers are generated by the generators of $C_{i,2}$ (see also the proof of Lemma 3.7.2). Hence, the quantity $B_i^{\mathsf{X}}(w)$ is equal to $|C_{i,2}(w)|$, the number of codewords in $C_{i,2}$ of weight w. This in turn is upper bounded by $|C_{i,1}(w)|$, where $C_{i,1}$ is the punctured Reed–Muller code. ($C_{i,2}$ is the even subcode of $C_{i,1}$.) Since $1^{n(i)} \notin C_{i,2}$ and so $|C_{i,2}(n(i))| = 0$, it suffices to show that

$$\max_{w\leqslant n(i)-1} \frac{\left|C_{i,1}(w)\right|}{\binom{n(i)}{w}} \leqslant \mathrm{negl}(n(i)). \tag{3.96}$$

First, note that since $1^{n(i)} \in C_{i,1}$, and so for every string $s \in C_{i,1}$ also $1^{n(i)} \oplus s \in C_{i,1}$, there are an equal amount of strings with weight w in the code as with weight $n(i)-w$. Because the denominator is also the same when exchanging w with $n(i)-w$, it suffices to consider only those $w \in \{1, \dots, (n(i)-1)/2\}$. We find an upper bound to the above expression, for three separate cases:

$0 < w < d(i)$: The bound follows directly from the error-correcting property of the code: there is no codeword in $C_{i,1}$ with weight less than the distance $d(i)$. Hence, $\left|C_{i,1}(w)\right| / \binom{n(i)}{w} = 0$ for these values of w.

$d \leqslant w < \frac{n(i)}{8}$: For this case, we can use the Ray-Chaudhuri-Wilson inequality (see Lemma 2.2.1). Consider two (nonidentical) strings $s, t \in C_{i,1}$ with $|s| = |t| = w$. Then, since $C_{i,1}$ is a linear code, if we define $u = s \oplus t$, we have that also $u \in C_{i,1}$. In particular, $|u| \geqslant d$. Now write $|u| = |s| + |t| - 2|s \wedge t| \geqslant d(i)$, so that we have

$$|s \wedge t| \leqslant \frac{1}{2}(|s| + |t| - d(i)) = w - \frac{d(i)}{2}. \tag{3.97}$$

Now, instead of bitstrings, we view all strings in $C_{i,1}$ of weight w as a family $\mathcal{F}_w$ of subsets of $[n(i)]$. To be more precise, define $\mathcal{F}_w = \left\{\{j \mid j \in [n(i)], s_j = 1\} \mid s \in C_{i,1}(w)\right\}$. We will bound the size of this set family, noting that $|\mathcal{F}_w| = |C_{i,1}(w)|$. From the previous argument, we have $\forall F, G \in \mathcal{F}_w, F \neq G : |F \cap G| \leqslant w - \frac{d(i)}{2}$.

From the Ray-Chaudhuri–Wilson inequality, it then immediately follows that

$$|\mathcal{F}_w| \leqslant \binom{n(i)}{w - \frac{d(i)}{2}}. \tag{3.98}$$

Now, it remains to analyze the ratio of this bound to the total number of strings of weight w.

$$\begin{aligned}
\frac{|\mathcal{F}_w|}{\binom{n(i)}{w}} &\leqslant \frac{\binom{n(i)}{w-\frac{d(i)}{2}}}{\binom{n(i)}{w}} && (3.99)\\
&= \frac{w!(n(i)-w)!}{(w-\frac{d(i)}{2})!(n(i)-w+\frac{d(i)}{2})!} && (3.100)\\
&= \frac{w^{\underline{\frac{d(i)}{2}}}}{(n(i)-w+\frac{d(i)}{2})^{\underline{\frac{d(i)}{2}}}} && (3.101)\\
&\leqslant \frac{\left(\frac{n(i)}{8}\right)^{\underline{\frac{d(i)}{2}}}}{\left(\frac{7n(i)}{8}\right)^{\underline{\frac{d(i)}{2}}}} && (3.102)\\
&\leqslant \left(\frac{1}{7}\right)^{\frac{d(i)}{2}}. && (3.103)
\end{aligned}$$

Here, $m^{\underline{k}}$ is the falling factorial.[7] The second-to-last inequality is by filling in the worst case for w (and dropping the additional $d(i)/2$ in the denominator). The final inequality follows by suitably grouping terms in the falling factorial and noting that $\frac{x-k}{y-k} \leqslant \frac{x}{y}$ for all $0 \leqslant x \leqslant y$ and $0 \leqslant k < y$.

Since for the Reed–Muller code the distance $d(i)$ is $\Omega(\sqrt{n(i)})$, this bound is negligible in $n(i)$.

$\frac{n(i)}{8} \leqslant w \leqslant \frac{n(i)-1}{2}$: For these weights we will compare the total number of elements of $C_{i,1}$ with $\binom{n(i)}{w}$, and show that $\frac{|C_{i,1}|}{\binom{n(i)}{w}} = \mathrm{negl}(n(i))$.

The total number of elements of $C_{i,1}$ is $2^{2^{2i}} = 2^{\frac{1}{2}(n(i)+1)}$. Without loss of generality, assume we look at $w = \frac{n(i)}{8}$, since the quantity we are computing, $\frac{|C_{i,1}|}{\binom{n(i)}{w}}$, is monotonically decreasing for all w in the range we are considering. Using general bounds on factorials ($\sqrt{2\pi}k^{k+1/2}e^{-k} \leqslant k! \leqslant ek^{k+1/2}e^{-k}$) to bound $\binom{n(i)}{n(i)/8}$,

[7] i.e., $m^{\underline{k}} = m(m-1)\cdots(m-k+1)$.

we can derive that

$$\frac{\left|C_{i,1}\right|}{\binom{n(i)}{w}} \leqslant \frac{e^2\sqrt{n(i)}}{\sqrt{2\pi}} \cdot 2^{\left(\frac{1}{8}+\frac{7}{8}\log_2\frac{7}{8}\right)n(i)+\left(\frac{1}{2}\log_2\frac{7}{8}-1\right)} \tag{3.104}$$

$$< 2^{-0.043n(i)+\frac{1}{2}\log_2(n(i))+1}, \tag{3.105}$$

which is negligible in $n(i)$.

Combining the cases in the above analysis, note that the maximum of $B_i^{\mathsf{X}}(w)/A_i^{\mathsf{X}}(w)$ over all $w \leqslant n(i)$ is upper bounded by the maximum of two negligible functions, which is itself negligible in $n(i)$. As an aside, we note that the upper bound for the second case in the analysis, $\left(\frac{1}{7}\right)^{\frac{d(i)}{2}}$, exceeds the upper bound for the third case whenever $i \geqslant 6$.

□

3.7.4 The strong trap code

With the terminology described in the previous subsections, we can now give a more formal definition of the strong trap code from Section 3.3.4. We define the code in terms of a parameter i, which is used to select the appropriate Reed–Muller code. For adequate security of the strong trap code, the parameter i should be chosen such that $n(i)$ is the desired security parameter.

Definition 3.7.8 (Strong trap code)**.** Let $(E_i)_{i\in\mathbb{N}}$ be a weight-sparse family of weakly self-dual CSS codes with parameters $[[n(i), 1, d(i) = \Omega(\sqrt{n(i)})]]$ and benign distance $\Omega(\sqrt{n(i)})$. Then the ith strong trap code $\{V_{i,k}\}_{k\in\mathcal{K}_i}$ encodes $m = 1$ qubit using $t = 3n(i) - 1$ tags with the unitaries $V_{i,k} := \pi_k E_i^{\otimes 3}\mathsf{H}_{2n(i)+1}$ (where $\mathsf{H}_{2n(i)+1} = \mathsf{I}^{\otimes 2n(i)} \otimes \mathsf{H} \otimes \mathsf{I}^{\otimes(n(i)-1)}$).

Theorem 3.7.9. *The strong trap code is a strong-purity-testing code with error negligible in $n(i)$.*

Proof. Consider an arbitrary i and nonidentity Pauli $P_\ell \in \mathscr{P}_{3n(i)} \setminus \{\mathsf{I}^{\otimes 3n(i)}\}$. Let w_x and w_z denote the X-weight and Z-weight (respectively) of P_ℓ, and note that their maximum $\max(w_x, w_z) > 0$.

We consider the probability (over k) that $P_{\ell'} := \pi_k^\dagger P_\ell \pi_k$ remains undetected by the code E_i and the traps. Because E_i is a CSS code, it detects X and Z errors separately: let us write $P_{\ell'} = P_x P_z$ with $P_x \in \{\mathsf{I},\mathsf{X}\}^{\otimes 3n(i)}$ and $P_z \in \{\mathsf{I},\mathsf{Z}\}^{\otimes 3n(i)}$, and focus first on the probability that P_x remains undetected, i.e., the probability that

$$\mathsf{H}_{2n(i)+1}(E_i^\dagger)^{\otimes 3} P_x E_i^{\otimes 3}\mathsf{H}_{2n(i)+1} \in \mathscr{P}_1 \otimes \{\mathsf{I},\mathsf{Z}\}^{\otimes 3n(i)-1}. \tag{3.106}$$

Because of the permutation π_k, P_x is a random Pauli in $\{\mathsf{I},\mathsf{X}\}^{\otimes 3n(i)}$ with weight w_x. (Note that P_z is also a random Pauli with weight w_z, but is correlated with P_x: any overlap in the locations of X and Z operators in P_ℓ is preserved by the permutation.)

Consider all possible values of $w_x = w_1 + w_2 + w_3$, where w_1 denotes the weight of P_x on the first (data) codeword, w_2 the weight on the second ($|0\rangle$-trap) codeword, and w_3 the weight on the third ($|+\rangle$-trap) codeword:

- If $w_x = 0$, then the Pauli P_x is identity, and remains undetected with probability 1.
- If $0 < w_x < d(i)$, then $0 < w_j < d(i)$ for at least one $j \in \{1,2,3\}$. E_i detects an error on the jth block with certainty, since the weight of the error is below the distance and the benign distance.
- If $d(i) \leqslant w_x \leqslant 3n(i) - d(i)$, the attack P_x will likely be detected on the second block, the $|0\rangle$-trap. We can be in one of four cases:
  - $w_2 > 0$ and P_x is detected in the second block by the CSS code E_i.
  - $w_2 > 0$ and P_x acts as a logical operation on the second block. Since P_x consists of only I's and X's, this logical operation can only be an X by the construction of CSS codes. In this case, P_x is detected by the projection that checks whether the trap is still in the $|0\rangle$ state.
  - $w_2 > 0$ and P_x acts as a stabilizer on the second block, and remains undetected on that block. However, by the weight-sparsity of the code family, the probability that this is the case is at most $f(i)$ (see Definition 3.7.4), which is negligible in $n(i)$.
  - $w_2 = 0$. In this case, P_x acts as identity on the second block. The probability that this case occurs, however, is small:

  $$\Pr_k[w_2 = 0] = \frac{\binom{2n(i)}{w_x}}{\binom{3n(i)}{w_x}} < \left(\frac{2}{3}\right)^{w_x} \leqslant \left(\frac{2}{3}\right)^{d(i)} . \tag{3.107}$$

  The first inequality holds in general for binomials, and the second one follows from the fact that $w_x \geqslant d(i)$. Since $d(i) = \Omega(\sqrt{n(i)})$, this probability is negligible in $n(i)$.

  In total, the probability of the attack remaining undetected for $d(i) \leqslant w_x \leqslant 3n(i) - d(i)$ is negligible in $n(i)$.
- If $3n(i) - d(i) < w_x < 3n(i)$: as in the second case, there is at least one $j \in \{1,2,3\}$ such that $n(i) - d(i) < w_j < n(i)$, causing the attack to be detected (recall that $\mathsf{X}^{\otimes 3n(i)}$ is a logical X, and therefore this mirrors the $0 < w_x < d(i)$ case).

- If $w_x = 3n(i)$, then the logical content of the second block, the $|0\rangle$-trap, is flipped. This is detected with certainty as well.

We see that unless $w_x = 0$, the Pauli P_x remains undetected only with probability at most

$$\max\left\{f(i), \left(\frac{2}{3}\right)^{d(i)}\right\}. \tag{3.108}$$

For a weight-sparse CSS family (i.e., $f(i)$ is negligible in $n(i)$) with distance $d(i) = \Omega(\sqrt{n(i)})$, this probability is negligible in $n(i)$.

A similar analysis can be made for P_z: it is always detected with high probability, unless $w_z = 0$. We stress that these probabilities are *not* independent. However, we can say that

$$\Pr_k[P_x \text{ and } P_z \text{ undetected}] \leqslant \min\left\{\Pr_k[P_x \text{ undetected}], \Pr_k[P_z \text{ undetected}]\right\}, \tag{3.109}$$

and since at least one of w_x and w_z is nonzero, this probability is negligible in $n(i)$. $\square$

When the trap code is instantiated with punctured Reed-Muller codes, then the weight-sparsity function $f(i) = \left(\frac{1}{7}\right)^{\frac{d(i)}{2}}$ whenever $i \geqslant 6$ (see Lemma 3.7.7). Therefore, for this choice of underlying code, the dominant term in the error calculation is simply $\left(\frac{2}{3}\right)^{d(i)}$. For example, if one wanted to construct a strong-purity-testing code with error $\varepsilon < 10^{-20}$ using this construction, one would need to use the 7th code in the code family, which requires codewords of $2^{15} - 1$ physical qubits.

3.8 Simultaneous encryptions with key reuse

Earlier work on key reuse for quantum authentication deals explicitly with *key recycling*, the decision to reuse (part of) a key for a new encryption after completing the transmission of some other quantum message. The key is reused only *after* the honest party decides whether to accept or reject the first message, so recycling is a strictly sequential setting.

If Construction 3.2.8 is instantiated with a strong-purity-testing code (such as the strong trap code), the resulting scheme is able to handle an even stronger *parallel* notion of key reuse. As long as the one-time pads are independent, it is possible to encrypt multiple qubits under the same code key while preserving security. Even if the adversary is allowed to interactively decrypt a portion of the qubits one-by-one, the other qubits will remain authenticated. This property is especially important for the strong trap code: computing on data authenticated with the strong trap code requires all qubits to be encrypted under the same permutation key.

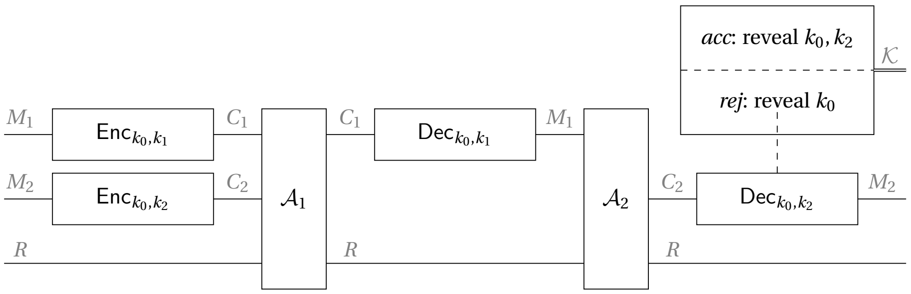


Figure 3.5: The real channel $\mathfrak{R}$. The "key reveal box" at the end corresponds to the key-recycling term in the real channel, the content of which depends on whether the scheme accepts or rejects.

The original trap code is secure in this setting (as long as the one-time pads are fresh [BGS13, Section 5.2]), but only if all qubits are decrypted at the same time. If some qubits can be decrypted separately, the adversary can sacrifice a few encoded qubits to apply the attack described in Section 3.3.2, and deduce the location of the traps. The authentication on the remaining qubits is then completely broken.

Suppose we encrypt two messages using an authentication scheme based on a strong-purity-testing code $\{V_{k_0}\}_{\mathcal{K}_0}$, using the same code key k_0 but a fresh one-time pad. If we then decrypt the first message, the scheme is still QCA-R-authenticating on the second message with only slightly worse security, as described by the following theorem. The argument easily extends to any polynomial number of authenticated qubits.

Theorem 3.8.1. *Let* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an* ε-QCA-R *scheme resulting from Construction 3.2.8, using a strong-purity-testing code* $\{V_{k_0}\}_{\mathcal{K}_0}$. *Let* M_1, M_2 *denote the plaintext registers of the two messages,* $C_1 = M_1T_1, C_2 = M_2T_2$ *the corresponding ciphertext registers, and* R *a side-information register. Let* $\mathcal{A}_1, \mathcal{A}_2$ *be arbitrary adversarial channels. Define the effective real channel* $\mathrm{Real}_{k_0,k_1,k_2}$, *for keys* k_0, k_1, k_2, *as*

$$\mathsf{Dec}_{k_0,k_2}^{C_2\to M_2} \circ \mathcal{A}_2^{M_1,C_2,R} \circ \mathsf{Dec}_{k_0,k_1}^{C_1\to M_1} \circ \mathcal{A}_1^{C_1,C_2,R} \circ \left(\mathsf{Enc}_{k_0,k_1}^{M_1\to C_1} \otimes \mathsf{Enc}_{k_0,k_2}^{M_2\to C_2}\right).$$

(See Figure 3.5 for a pictorial representation.) Define the channels Π_{acc} *and* Π_{rej} *as projecting on the accepting and rejecting outcomes of the decryption procedure, respectively. That is,*

$$\Pi_{\mathsf{acc}}^{M} : \rho \mapsto \left(\mathbb{1}^M - |\bot\rangle\langle\bot|^M\right)\rho\left(\mathbb{1}^M - |\bot\rangle\langle\bot|^M\right)^\dagger$$
$$\Pi_{\mathsf{rej}}^{M} : \rho \mapsto |\bot\rangle\langle\bot|^M \rho\left(|\bot\rangle\langle\bot|^M\right)^\dagger.$$

Then there exist a simulator $\mathcal{S} = \mathcal{S}_{\mathsf{acc}} + \mathcal{S}_{\mathsf{rej}}$ where $\mathcal{S}_{\mathsf{acc}}$ is as in Definition 3.2.4, and $\mathcal{S}_{\mathsf{rej}}$ is such that the key-recycling real channel

$$\begin{aligned}\mathfrak{R} : \rho^{M_1 M_2 R} \mapsto \mathbb{E}_{k_0,k_1,k_2}\Big[&\Big(\Pi^{M_2}_{\mathsf{acc}} \circ \mathrm{Real}_{k_0,k_1,k_2}(\rho)\Big) \otimes |k_0,k_2\rangle\langle k_0,k_2| \\ &+ \Big(\Pi^{M_2}_{\mathsf{rej}} \circ \mathrm{Real}_{k_0,k_1,k_2}(\rho)\Big) \otimes |k_0\rangle\langle k_0|\Big]\end{aligned}$$

is 2ε-close in diamond-norm distance to the ideal channel,

$$\mathfrak{I} : \mathbb{I}^{M_2} \otimes \mathcal{S}_{\mathsf{acc}} \otimes \mathbb{E}_{k_0,k_2}\left[|k_0,k_2\rangle\langle k_0,k_2|\right] \;+\; |\bot\rangle\langle\bot|^{M_2}\,\mathrm{Tr}_{M_2} \otimes\, \mathcal{S}_{\mathsf{rej}} \otimes \mathbb{E}_{k_0}\left[|k_0\rangle\langle k_0|\right].$$

*That is, the scheme is 2ε-*QCA-R*-authenticating on the second qubit.*

Proof sketch. As a first step, we rewrite the encryption of the second qubit as using encoding and teleportation, by using the equivalence between applying a random quantum one-time pad and teleporting a state (see Section 2.4.7). The encryption of the second qubit can then be thought of as happening after decryption of the first qubit. Next, we apply QCA-R security of the first qubit, where we are using the property that k_0 is recycled both in the accept and the reject case. Finally we undo the rewrite and can directly apply QCA-R security on the remaining state. □

Proof. First, note that the one-time-pad key k_1 is picked completely at random, used only for the encryption and decryption of the first qubit. In particular, in the final situation as represented by $\mathfrak{R}$ we do not require key recycling for k_1.

A quantum one-time pad that uses a uniformly random key is completely equivalent to a teleportation. Therefore, in the real channel $\mathfrak{R}$, it does not matter whether the key k_2 is picked randomly beforehand, or whether it is the random outcome $\hat{k}_2$ of a teleportation measurement. We denote the channel where the key of the second qubit comes from teleportation by $\mathfrak{R}_2$ (see Figure 3.6). Because this rewriting doesn't change functionality, we have that $\|\mathfrak{R}_2 - \mathfrak{R}\|_\diamond = 0$.

While Figure 3.6 likely provides the clearest description of $\mathfrak{R}_2$, we also write out its symbolic definition below for completeness.

Let $\Gamma^{EC_2 \to \hat{\mathcal{K}}_2}$ be the quantum channel that performs a pairwise Bell measurement between the n qubits in register E and the n qubits in C_2, and stores the $2n$ classical outcome bits as the "key" $\hat{k}_2$. That is, let

$$|\Phi_k\rangle_{EC_2} := \left(\bigotimes_{i=1}^{n} \mathsf{X}^{k_i}\right)^{E} \left(\bigotimes_{i=n+1}^{2n} \mathsf{Z}^{k_i}\right)^{E} \bigotimes_{i=1}^{n} |\Phi^+\rangle^{(E)_i (C_2)_i} \tag{3.110}$$

be the basis of all possible n-qubit Bell states between pairwise qubits of E and C_2. The measurement channel is then defined by the operation

$$\Gamma^{EC_2 \to \hat{\mathcal{K}}_2}(\rho) := \sum_{\hat{k}_2} |\hat{k}_2\rangle\langle\hat{k}_2| \left\langle \Phi_{\hat{k}_2}\right|^{EC_2} \rho^{EC_2} \left|\Phi_{\hat{k}_2}\right\rangle^{EC_2}. \tag{3.111}$$

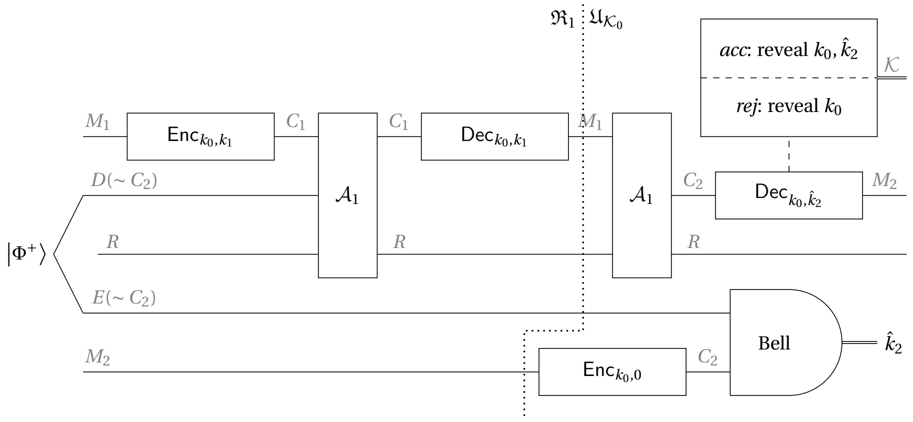


Figure 3.6: The channel $\mathfrak{R}_2$, which is equivalent to $\mathfrak{R}$. The encryption of the second qubit is done by first encrypting it without a one-time pad (key 0), and then teleporting it through a set of EPR pairs $|\Phi^+\rangle$, yielding the key $\hat{k}_2$. We separate the channel into two parts, $\mathfrak{R}_1$ and $\mathfrak{U}$.

Define the shorthand:

$$\begin{aligned}\mathrm{Real}'^{M_1M_2R\to M_2R\hat{\mathcal{K}}_2}_{k_0,k_1} &:= \mathsf{Dec}^{C_2\to M_2}_{k_0,\hat{\mathcal{K}}_2}\circ\mathcal{A}_2^{M_1,C_2,R}\circ\mathsf{Dec}^{C_1\to M_1}_{k_0,k_1}\circ\\ &\left(\mathcal{A}_1^{C_1,D,R}\otimes\Gamma^{EC_2\to\hat{\mathcal{K}}_2}\right)\circ\left(\mathsf{Enc}^{M_1\to C_1}_{k_0,k_1}\otimes\mathsf{Enc}^{M_2\to C_2}_{k_0,0}\otimes|\Phi^+\rangle\langle\Phi^+|^{DE}\right), \end{aligned}\tag{3.112}$$

where we slightly abuse notation in letting the final Dec use a copy of the contents of the $\hat{\mathcal{K}}_2$ register as a key. The rewritten channel becomes

$$\begin{aligned}\mathfrak{R}_2:\rho^{M_1M_2R}\mapsto\mathbb{E}_{k_0,k_1}\Big[&\left(\Pi^{M_2}_{\mathsf{acc}}\circ\mathrm{Real}'^{M_1M_2R\to M_2R\hat{\mathcal{K}}_2}_{k_0,k_1}(\rho)\right)\otimes|k_0\rangle\langle k_0|\\ &+\left(\mathrm{Tr}_{\hat{\mathcal{K}}_2}\Pi^{M_2}_{\mathsf{rej}}\circ\mathrm{Real}'^{M_1M_2R\to M_2R\hat{\mathcal{K}}_2}_{k_0,k_1}(\rho)\right)\otimes|k_0\rangle\langle k_0|\Big].\end{aligned}\tag{3.113}$$

(In the accept branch, the $\Gamma^{EC_2\to\hat{\mathcal{K}}_2}$ outputs the register $\hat{\mathcal{K}}_2$, so that it is output as part of the recycled key.) This concludes the formal definition of the channel $\mathfrak{R}_2$.

Our next step will be to apply the QCA-R security of the first qubit. Note that none of the registers $M_2, E, \hat{\mathcal{K}}_2$ are used by $\mathcal{A}_1$ in the channel $\mathfrak{R}_2$. Thus, we can freely shift the timing of the encryption and Bell measurement of the second qubit around, and make them occur only *after* decryption of the first qubit. Doing so allows us to view $\mathfrak{R}_2$ as a concatenation of two channels (see Figure 3.6), where the first ($\mathfrak{R}_1$) only depends on k_0, k_1, and the second ($\mathfrak{U}_{\mathcal{K}_0}$) on k_0, k_2.

More formally, define $\mathfrak{R}_1$ as the real channel (as in Definition 3.2.4) of adversary $\mathcal{A}_1$ acting on qubit 1. That is, $\mathfrak{R}_1^{M_1DR\to M_1R\mathcal{K}_0\mathcal{K}_1}$ equals the map

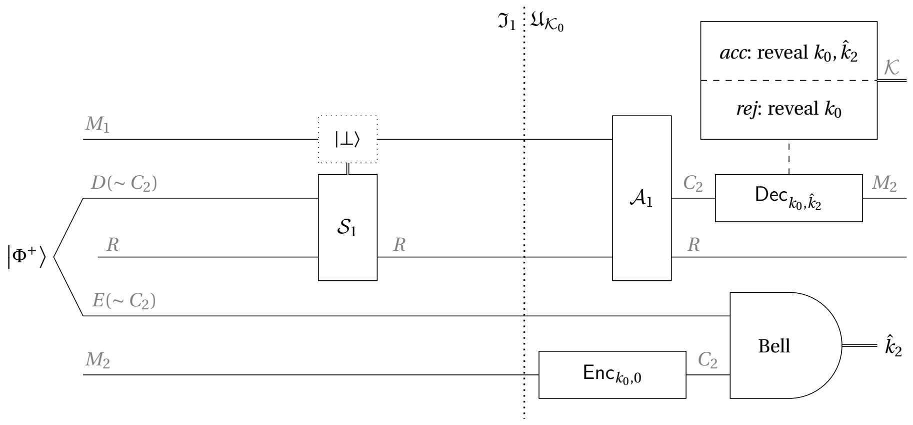


Figure 3.7: The channel $\mathfrak{R}_3$, where the real channel $\mathfrak{R}_1$ is replaced by the ideal channel $\mathfrak{I}_1$. The dotted box represents the conditional replacement of M_1 by the state $|\bot\rangle\langle\bot|$, in the reject case of the simulator $\mathcal{S}_1$.

$$\rho \mapsto \mathop{\mathbb{E}}_{k_0,k_1}\Big[\Big(\Pi^{M_1}_{\mathsf{acc}} \otimes \mathsf{Dec}^{C_1\to M_1}_{k_0,k_1} \circ \mathcal{A}_1^{C_1,D,R} \circ \mathsf{Enc}^{M_1\to C_1}_{k_0,k_1}\Big)(\rho) \otimes |k_0,k_1\rangle\langle k_0,k_1| \\ + \Big(\Pi^{M_1}_{\mathsf{rej}} \otimes \mathsf{Dec}^{C_1\to M_1}_{k_0,k_1} \circ \mathcal{A}_1^{C_1,D,R} \circ \mathsf{Enc}^{M_1\to C_1}_{k_0,k_1}\Big)(\rho) \otimes |k_0\rangle\langle k_0|\Big]. \tag{3.114}$$

Then we can write

$$\mathfrak{R}_2(\rho) = \mathfrak{U}_{\mathcal{K}_0} \circ (\mathrm{Tr}_{\mathcal{K}_1}\mathfrak{R}_1^{M_1DR} \otimes \mathbb{1}^{E,C_2})(\rho^{M_1,M_2,R} \otimes |\Phi^+\rangle\langle\Phi^+|^{DE}), \tag{3.115}$$

where we emphasize that the key k_1 is not revealed to the second channel $\mathfrak{U}_{\mathcal{K}_0}$.

By QCA-R security of the scheme, we know that there exists a simulator $\mathcal{S}_1^{D,R} = \mathcal{S}^1_{\mathsf{acc}} + \mathcal{S}^1_{\mathsf{rej}}$ such that the corresponding ideal channel $\mathfrak{I}_1$ is ε-close to $\mathfrak{R}_1$ in the diamond norm. The ideal channel $\mathfrak{I}_1^{M_1DR\to M_1R\mathcal{K}_0\mathcal{K}_1}$ is given by

$$\rho \mapsto \big(\mathbb{1}^{M_1} \otimes \mathcal{S}^1_{\mathsf{acc}}\big)\big(\rho \otimes \mathbb{E}_{k_0,k_1}[|k_0,k_1\rangle\langle k_0,k_1|]\big) \\ + \Big(|\bot\rangle\langle\bot|^{M_1}\,\mathrm{Tr}_{M_1} \otimes \mathcal{S}^1_{\mathsf{rej}}\Big)\big(\rho \otimes \mathbb{E}_{k_0}[|k_0\rangle\langle k_0|]\big). \tag{3.116}$$

Within $\mathfrak{R}_2$, we can replace $\mathfrak{R}_1$ by its ideal channel $\mathfrak{I}_1$. Call the resulting channel $\mathfrak{R}_3$ (see Figure 3.7). Since $\frac{1}{2}\|\mathfrak{R}_1 - \mathfrak{I}_1\|_\diamond \leq \varepsilon$, and the distance between any two channels is monotonically nonincreasing when concatenated with another quantum operation, we have that $\frac{1}{2}\|\mathfrak{R}_2 - \mathfrak{R}_3\|_\diamond \leq \varepsilon$.

By replacing the real channel on the first qubit by the ideal one, we have effectively gotten rid of any prior dependence of key k_0. We can now shift back from the

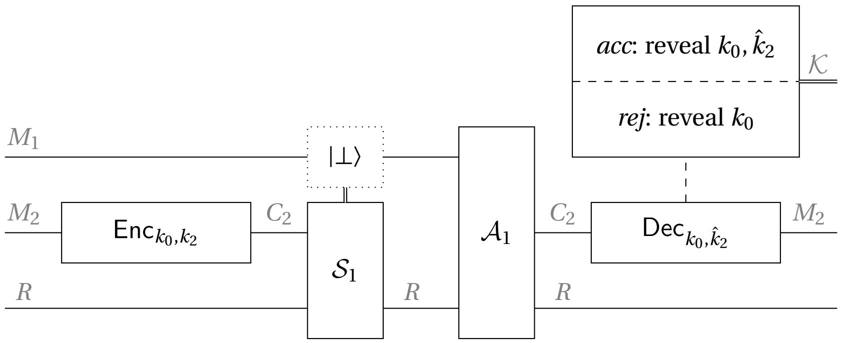


Figure 3.8: An equivalent representation of the channel $\mathfrak{R}_3$, where the encryption via teleportation is replaced with a regular encryption map for the second qubit.

teleportation point-of-view, and encrypt the second qubit at the start of the circuit again (see Figure 3.8). Doing so does not change the functionality of the channel.

On the channel depicted in Figure 3.8, we can apply the security of QCA-R again, with as adversarial map

$$\mathcal{A}_2 \circ (\mathbb{1}^{M_1} \otimes \mathcal{S}^1_{\mathsf{acc}} + |\bot\rangle\langle\bot|^{M_1} \operatorname{Tr}_{M_1} \otimes \mathcal{S}^1_{\mathsf{rej}}). \tag{3.117}$$

Applying the security definition gives rise to an ideal channel $\mathfrak{I}$ such that $\frac{1}{2}\|\mathfrak{I} - \mathfrak{R}_3\|_\diamond \leqslant \varepsilon$.

Earlier on, we established that $\frac{1}{2}\|\mathfrak{R} - \mathfrak{R}_2\|_\diamond = 0$, and $\frac{1}{2}\|\mathfrak{R}_2 - \mathfrak{R}_3\|_\diamond \leqslant \varepsilon$. By triangle inequality, the original real channel $\mathfrak{R}$ is at most 2ε from the ideal channel $\mathfrak{I}$. □

3.9 Conclusion

We presented a new security definition, QCA-R, for ciphertext authentication with key recycling, and showed that schemes based on purity-testing codes satisfy quantum ciphertext authentication, while strong purity testing implies both ciphertext authentication and key recycling. This is analogous to the security of quantum plaintext-authentication schemes from purity-testing codes [BCG+02; Por17].

Additionally, we constructed the *strong trap code*, a variant of the trap code which is a strong-purity-testing code and therefore is QCA-R secure (as well as secure under all notions of plaintext authentication). This new scheme can strengthen security and add key recycling to earlier applications of the trap code. It is also applicable in a wider range of applications than the original trap code, because encrypted qubits remain secure even if other qubits sharing the same key are decrypted earlier.

A potential application of the strong trap code is the design of a quantum CCA2-secure encryption scheme [AGM18, Definition 9] that allows for computation on the encrypted data. By only using the pseudo-random generator for the one-time-pad keys, and recycling the key for the underlying error-correcting code, this security level could be achieved.

3.9.1 Future directions

As future work, our definition of QCA-R could be generalized in different ways. First, one can consider a variant of the definition in the abstract-cryptography or universal-composability framework, in order to ease the composition with other cryptographic primitives. Second, because it can be useful to authenticate measurements in delegated-computation applications, one could extend the definition of QCA-R to deal with the measurement of authenticated data. We expect no real obstacles for this extension of the definition, and refer to comparable work on the original trap code for guidance [BGS13, Appendix B.2].

Quantum message authentication can also be explored in a much broader sense. The notions described in this chapter are all forms of *one-time* authentication, where security is guaranteed as long as the key is used only once. In Section 3.8, we saw that for codes based on strong purity testing, part of the key could be used several times, simultaneously, as long as the one-time-pad part of the key was refreshed. It may be possible to adapt the code to allow reusing the entire key. Because of the relation between quantum authentication and quantum encryption, such codes would likely require computational assumptions: for example, one could use a classical authenticated encryption scheme to encrypt the pad keys. We take this approach in Chapter 5 in the context of homomorphic encryption, but only consider security in a single round.

Another direction to explore could be authentication under the presence of noise. In this work, we think of our communication channels and computation steps as being perfectly noiseless, but in a real-world application this will not be the case. To be able to verify authenticated messages in such situations, one can add an error-correcting code on top of the authentication code. However, it would be more efficient to use the error-correcting properties inside the quantum authentication code directly. The receiver may decide to accept a message if only a small amount (or a certain pattern) of error occurs in the tag state, since that could be attributed to natural noise. One would have to investigate whether such a change to the protocols would still yield sufficiently secure authentication.

4
Multi-party quantum computation

Chapter contents

4.1 Introduction

In secure multi-party computation (MPC), two or more players want to jointly compute some publicly known function on their private data, without revealing their inputs to the other players. Since its introduction by Yao [Yao82], MPC has been extensively developed in different setups, leading to applications of both theoretical and practical interest (see, e.g., [CDN15] for a detailed overview).

With the emergence of quantum technologies, it becomes necessary to understand its consequences in the field of MPC. First, classical MPC protocols have to be secured against quantum attacks. But also, the increasing number of applications where quantum computational power is desired motivates protocols enabling multi-party *quantum* computation (MPQC) on the players' private (possibly quantum) data. In this chapter, we focus on the second task.

MPQC was first studied by Crépeau, Gottesman and Smith [CGS02], who proposed a k-party protocol based on verifiable secret sharing that is information-theoretically secure, but requires the assumption that at most $k/6$ players are dishonest. The fraction $k/6$ was subsequently improved to $< k/2$ [BCG+06] which is optimal for secret-sharing-based protocols due to no-cloning. The case of a dishonest majority was thus far only considered for $k = 2$ parties, where one of the two players can be dishonest [DNS10; DNS12; KMW17][1]. These protocols are based on different cryptographic techniques, in particular quantum authentication codes in conjunction with classical MPC [DNS10; DNS12] or quantum-secure bit commitment and oblivious transfer [KMW17]. In an authentication-based protocol, the players encode their inputs using a quantum authentication code to prevent the other, potentially adversarial, players from making unauthorized alterations to their data. That way, they can ensure that the output of the computation is in the correct logical state.

In this chapter, we propose the first secure MPQC protocol for any number k of players in the dishonest majority setting, i.e., the case with up to $k-1$ colluding adversarial players. (In the case where there are k adversaries and no honest players, there is nobody whose input privacy and output authenticity is worth protecting.) Our protocol builds on the authentication-based two-party protocol of Dupuis, Nielsen, and Salvail [DNS12], which we now describe in brief.

Two-party quantum computation. The DNS protocol uses a classical MPC protocol, and involves two parties, Alice and Bob, of whom at least one is honestly following the protocol. Alice and Bob encode their inputs using a technique called *swaddling*: if Alice has an input qubit $|\psi\rangle$, she first encodes it using the n-qubit Clifford code (see Section 3.3.1), resulting in $A\left(|0^n\rangle \otimes |\psi\rangle\right)$, for some random $(n+1)$-qubit Clifford A sampled by Alice, where n is the security parameter. Then, she sends the state to

[1]Kashefi and Pappa [KP17] consider an asymmetric setting where the protocol is secure only when some specific sets of $k-1$ players are dishonest.

Bob, who puts another encoding on top of Alice's: he creates the "swaddled" state $B\left(A\left(|0^n\rangle \otimes |\psi\rangle\right) \otimes |0^n\rangle\right)$ for some random $(2n+1)$-qubit Clifford B sampled by Bob. This encoded state consists of $2n+1$ qubits, and the data qubit $|\psi\rangle$ sits in the middle.

If Bob wants to test the state at some point during the protocol, he simply needs to undo the Clifford B, and test that the last n traps are $|0\rangle$. However, if Alice wants to test the state, she needs to work together with Bob to access her traps. Using classical multi-party computation, they jointly sample a random $(n+1)$-qubit Clifford B' which is only revealed to Bob, and compute a Clifford $C := (\mathsf{I}^{\otimes n} \otimes B')(A^\dagger \otimes \mathsf{I}^{\otimes n})B^\dagger$ that is only revealed to Alice. Alice, who will not learn any relevant information about B or B', can use C to "flip" the swaddle, revealing her n trap qubits for measurement. After checking that the first n qubits are $|0\rangle$, she adds a fresh $(2n+1)$-qubit Clifford on top of the state to re-encode the state, before computation can continue.

Single-qubit Clifford gates are performed simply by classically updating the inner key: if a state is encrypted with Cliffords BA, updating the decryption key to $BAG^\dagger$ effectively applies the gate G. See also Section 3.4 for a detailed discussion on how to compute on Clifford-authenticated data. In order to avoid that the player holding the inner key A skips the step of updating it to $AG^\dagger$, both players keep track of their keys using a classical commitment scheme. This can be encapsulated in the classical MPC, which we can assume acts as a trusted third party with a memory [BCG+06].

CNOT operations and measurements are slightly more involved, and require both players to test the authenticity of the relevant states several times. Hence, the communication complexity scales linearly with the number of CNOTs and measurements in the circuit.

Finally, to perform T gates, the DNS protocol makes use of magic states (see Section 2.4.4). To obtain reliable magic states, Alice generates a large number of them, so that Bob can test a sufficiently large fraction. He decodes them (with Alice's help), and measures whether they are in the expected state. If all measurements succeed, Bob can be sufficiently certain that the untested (but still encoded) magic states are in the correct state as well.

Extension to multi-party quantum computation. A natural question is how to lift a two-party quantum computation protocol to a multi-party quantum computation protocol. We discuss some of the issues that arise from two straightforward approaches, making them either infeasible or inefficient.

The first naive idea is trying to split the k players in two groups and make the groups simulate the players of any two-party protocol, whereas internally, the players run $k/2$-party computation protocols for all steps in the two-party protocol. Those $k/2$-party protocols are in turn realized by running $k/4$-party protocols, et cetera, until at the lowest level, the players can run actual two-party protocols.

Trying to construct such a composition in a black-box way, using the *ideal functionality* of a two-party protocol, one immediately faces a problem: at the lower levels, players learn intermediate states of the circuit, because they receive plaintext outputs from the ideal two-party subfunctionality. This immediately breaks the privacy of the protocol. If, on the other hand, we required the ideal two-party functionality to output encoded states instead of plaintexts, the size of the ciphertext would grow at each level. The overhead of this approach would be $O(n^{\log k})$, where $n \geqslant k$ is the security parameter of the encoding: that overhead is superpolynomial in the number of players.

Another idea is to extend the two-party protocol [DNS12] to multiple parties by adapting the subprotocols to work for more than two players. While this approach likely leads to a correct and secure protocol for k parties, the computational costs of such an extension could be high. First, note that in such an extension, each party needs to append n trap qubits to the encoding of each qubit, causing an overhead in the ciphertext size that is linear in k. Secondly, in this naive extension, the players need to create $\Theta(2^k)$ magic states for T gates, since each party would need to sequentially test at least half of the ones approved by all previous players.

4.1.1 Contributions

In this chapter, we present a protocol for multi-party quantum computation that is actively secure against up to $k-1$ actively colluding adversaries. Our protocol builds on the work of Dupuis, Nielsen, and Salvail [DNS10; DNS12], and like it, assumes a classical MPC, and achieves the same security guarantees as this classical MPC. In contrast to the naive extension described above, the quantum round complexity of our protocol for the computation of a circuit of $\{\mathsf{CNOT}, \mathsf{T}\}$ depth d is $O(k(d+\log n))$.

We remark that we achieve *composable security*, which is proven according to the standard ideal-vs.-real definition (see Section 2.3.3). Like other authentication-based protocols [DNS10; DNS12], our protocol assumes a classical MPC that is secure against a dishonest majority, and achieves the same security guarantees as this classical MPC. In particular, if we instantiate this classical MPC with an MPC in the *pre-processing model* [BDOZ11; DPSZ12; KPR18; CDE+18], our construction yields an MPQC protocol consisting of a classical "offline" phase used to produce authenticated shared randomness among the players, and a second "computation" phase, consisting of our protocol, combined with the "computation" phase of the classical MPC. The security of the "offline" phase requires computational assumptions, but assuming no attack was successful in this phase, the second phase has information-theoretic security.

In order to achieve our result, we make two major alterations to the two-party protocol of Dupuis et al. [DNS12] to efficiently extend it to a general k-party protocol. We briefly describe them below.

Public authentication test. In the two-party protocol, given a security parameter n, each party adds n qubits in the state $|0\rangle$ to each input qubit in order to authenticate it.

The size of each ciphertext is thus $2n+1$. The extra qubits serve as check qubits (or "traps") for each party, which can be measured at regular intervals: if they are nonzero, somebody tampered with the state.

In a straightforward generalization to k parties, the ciphertext size would become $kn+1$ per input qubit, putting a strain on the computing space of each player. In our protocol, the ciphertext size is constant in the number of players: it is usually $n+1$ per input qubit, temporarily increasing to $2n+1$ for qubits that are involved in a computation step. As an additional advantage, our protocol does not require that all players measure their traps every time a state needs to be checked for its authenticity.

To achieve this smaller ciphertext size, we introduce a *public authentication test*. Our protocol uses a single, shared set of traps for each qubit. If the protocol calls for the authentication to be checked, only the player that currently holds the state has to measure them. Of course, she cannot be trusted to simply honestly measure those traps. Instead, she temporarily adds extra trap qubits, and fills them with an encrypted version of the content of the existing traps. Now she measures only the newly created ones. The encryption ensures that the measuring player does not know the expected measurement outcome. If she is dishonest and has tampered with the state, she would have to guess a random n-bit string, or be detected by the other players. We design a similar test that checks whether a player has honestly created the first set of traps for their input at encoding time.

Efficient magic-state preparation. For the computation of non-Clifford gates, the protocol requires the existence of authenticated "magic states", auxiliary qubits in a known and fixed state that aid in the computation. In a two-party setting, one of the players can create a large number of such states, and the other player can, if he distrusts the first player, test a random subset of them to check if they were honestly initialized. Those tested states are discarded, and the remaining states are used in the computation.

In a k-party setting, such a "cut-and-choose" strategy where all players want to test a sufficient number of states would require the first party to prepare an exponential number (in k) of authenticated magic states, which quickly gets infeasible as the number of players grows. Instead, we need a testing strategy where dishonest players have no control over which states are selected for testing. We ask the first player to create a polynomial number of authenticated magic states. Subsequently, we use classical MPC to sample random, disjoint subsets of the proposed magic states, one for each player. Each player continues to decrypt and test their subset of states. The random selection process implies that, conditioned on the test of the honest player(s) being successful, the remaining registers indeed contain encrypted states that are reasonably close to magic states. Finally, we use standard magic-state distillation to obtain auxiliary inputs that are exponentially close to magic states.

In Section 4.2, we discuss the precise definitions of MPC and MPQC. In Section 4.3, we give a basic overview of our protocol, and describe a technical tool that we will use throughout this chapter. Section 4.4 describes details of the setup phase, including input encoding. Sections 4.5 and 4.6 give explicit protocols for Clifford computation and Clifford+T computation, respectively, and prove their security.

4.2 Definitions

4.2.1 Classical multi-party computation

In k-party computation, each player $i \in [k]$ holds a secret input x_i. The goal is to compute a function (represented by a circuit) on the input $(x_1, x_2 \ldots, x_k)$. A trivial way to do so is for all players to send their inputs to one central player, who computes the output and announces it. However, the players want to keep their individual inputs private, and furthermore they do not necessarily trust each other to perform an honest computation. We thus have to employ more clever techniques in order to achieve MPC functionality.

There are many different possible adversarial models [CDN15], so we only discuss the main types here. In all models, we assume that there is a single adversary that “corrupts” a number of players: behind the scenes, that single adversary replaces all those players, sees all of their incoming communication, and can use all of that combined information to produce their outgoing communication. Thus, when talking about a set of dishonest players, one may think of those players as actively sharing all their knowledge between themselves.

The first main distinction in types of adversaries is *passive* versus *active*. A passive adversary can see the inputs of the players it corrupted, and listens in on their ingoing and outgoing communication wires. It tries to learn as much information as possible (about the other players' inputs) as it can while doing so, but it cannot influence the internal computations of the corrupted players. An *active* adversary, on the other hand, is allowed to completely take over the corrupted players' computations, and actively deviate from the protocol, sending false messages in an attempt to learn information or to sabotage the computation. For convenience, we will assume that an active adversary communicates messages of appropriate sizes to honest players at appropriate times during the protocol; if it does not, it is of course easy to detect that adversarial action is going on. In the quantum setting, there is an intermediate type called a *specious* adversary, who can deviate from the protocol, but has to be able to revert the corrupted players' internal memory to an “honest” state at any time [DNS10].

The second distinction in types of adversaries is *static* versus *dynamic* (or *adaptive*). A static adversary chooses which players to corrupt before the start of the protocol, and that choice is fixed. A dynamic adversary, on the other hand, can choose to corrupt additional adversaries during the protocol. This choice can be based on any information the adversary has access to at that point in the protocol. In this work, we will consider static adversaries, but note that if an adversary can corrupt all-but-one player, it does not matter whether they do so at the start, or progressively throughout the protocol.

Informally, we say that an MPC protocol is secure if the following two properties hold: (1) the adversary gains no information about the honest players' private inputs, and (2) if the players do not abort the protocol, then at the end of the protocol they share a state corresponding to the correct computation applied to the inputs of honest players, and some choice of inputs for the dishonest (corrupted) players.

For multi-party computations where possibly more than half of the players are corrupted by the adversary, the possibility of an "abort" in point 2 is necessary: Cleve has shown [Cle86] that an MPC protocol against a dishonest majority cannot provide *fairness*, which asks that either all parties receive the protocol output or nobody does. In that setting, a dishonest player can always abort the protocol at any point, for example after having learned an unfavorable outcome of the protocol, before the honest player(s) have obtained their output(s). Hence, we have to settle for protocols allowing abort.

The formal security of MPC is captured by an ideal functionality, from which an actual protocol should be indistinguishable from the point of view of an environment. Throughout this chapter, we will utilize the following ideal MPC functionality as a black box:

Definition 4.2.1 (Ideal classical k-party stateful computation with abort)**.** Let $f_1, \dots, f_k$ and f_S be public classical deterministic functions on $k+2$ inputs. Let a string s represent the internal state of the ideal functionality. (The first time the ideal functionality is called, s is empty.) Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.

1. Every player $i \in [k]$ chooses an input x_i of appropriate size, and sends it (securely) to a trusted third party.
2. The trusted third party samples a bit string r uniformly at random.
3. The trusted third party computes $f_i(s, x_1, \dots, x_k, r)$ for all $i \in [k] \cup \{S\}$.
4. For all $i \in I_{\mathcal{A}}$, the trusted third party sends $f_i(s, x_1, \dots, x_k, r)$ to player i.
5. All $i \in I_{\mathcal{A}}$ respond with a bit b_i, which is 1 if they choose to abort, or 0 otherwise.
6. If $b_j = 0$ for all j, the trusted third party sends $f_i(s, x_1, \dots, x_k, r)$ to the other players $i \in [k] \backslash \mathcal{A}$ and stores $f_S(s, x_1, \dots, x_k, r)$ in an internal state register (replacing s). Otherwise, he sends an `abort` message to those players.

A few remarks about Definition 4.2.1 are in order. First, each player i gets an individual outcome determined by the function f_i. This choice provides more flexibility than computing the same function f for everyone: for example, we can reveal the outcome of some computation to only a subset of players (by fixing $f_i = 0$ for all other players), or we can give every player a different random string. Note that the description of f_i is not secret for the other players: only the input x_i is.

Second, the functions f_i are described as deterministic functions, taking a random string r as input. We do this in order to facilitate shared randomness: by supplying a common reference string, the outcomes of, e.g., f_1 and f_2 can be random but correlated. We do not specify an exact length for the string r, but we assume that it is sufficiently long to supply both private and public randomness for the execution on all functions f_i.

Third, the ideal functionality has *memory* that can carry over through multiple rounds of computation. The memory is represented by a string s, and at each round, can be updated as a function f_S of the previous memory, all inputs, and the randomness. In this work, we will mainly use the memory to store authentication keys: at the beginning of the protocol, secret keys for the authentication of the players' inputs are stored inside the memory, so that they may be updated throughout the computation, and retrieved at decryption time.

A concrete protocol is a (computationally) secure k-party computation protocol with abort if, informally, its execution is (quantum computationally) indistinguishable from the ideal functionality described in Definition 4.2.1, for all sets of (quantum) adversaries $I_{\mathcal{A}}$ (cf. Definition 4.2.3 for the formal definition in the quantum setting). Although many protocols exist for multi-party computation, each with their own strengths and weaknesses, only recently a post-quantum secure protocol that meets the requirements from Definition 4.2.1 was developed [ABG+20]. Its security relies on the hardness of the learning-with-errors problem.

4.2.2 Quantum multi-party computation

In this subsection, we describe the ideal functionality we aim to achieve for multi-party *quantum* computation (MPQC) with a dishonest majority. As noted in Section 4.2.1, we cannot hope to achieve fairness: therefore, we consider an ideal functionality with the option for the dishonest players to abort.

Definition 4.2.2 (Ideal quantum k-party computation with abort)**.** Let C be a quantum circuit on $W \in \mathbb{N}_{>0}$ wires. Consider a partition of the wires into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.

1. Every player $i \in [k]$ sends the content of R_i^{in} to the trusted third party.

2. The trusted third party populates R^{ancilla} with computational-zero states.

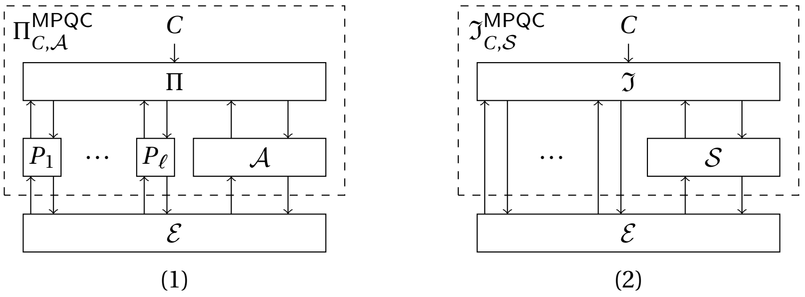


Figure 4.1: (1) The environment interacting with the protocol as run by honest players $P_1,\dots,P_\ell$, and an adversary who has corrupted the remaining players. (2) The environment interacting with a simulator running the ideal functionality.

3. The trusted third party applies the quantum circuit C on the wires $[W]$.
4. For all $i \in I_{\mathcal{A}}$, the trusted third party sends the content of R_i^{out} to player i.
5. All $i \in I_{\mathcal{A}}$ respond with a bit b_i, which is 1 if they choose to abort, or 0 otherwise.
6. If $b_i = 0$ for all i, the trusted third party sends the content of R_i^{out} to the other players $i \in [k] \backslash I_{\mathcal{A}}$. Otherwise, he sends an `abort` message to those players.

In Definition 4.2.2, all corrupted players individually choose whether to abort the protocol (and thereby to prevent the honest players from receiving their respective outputs). In reality, however, one cannot prevent several corrupted players from actively working together and sharing all information they have among each other. To ensure that our protocol is also secure in those scenarios, we consider security against a general adversary that corrupts all players in $I_{\mathcal{A}}$, by replacing their protocols by a single (interactive) algorithm $\mathcal{A}$ that receives the registers $R_{\mathcal{A}}^{\mathsf{in}} := R \sqcup \bigsqcup_{i \in I_{\mathcal{A}}} R_i^{\mathsf{in}}$ as input, and after the protocol produces output in the register $R_{\mathcal{A}}^{\mathsf{out}} := R \sqcup \bigsqcup_{i \in I_{\mathcal{A}}} R_i^{\mathsf{out}}$. Here, R is a side-information register in which the adversary may output extra information.

We will always consider protocols that fulfill the ideal functionality with respect to some gate set $\mathcal{G}$: the protocol should then mimic the ideal functionality only for circuits C that consist of gates from $\mathcal{G}$. This security is captured by the definition below.

Definition 4.2.3 (Computational security of quantum k-party computation with abort)**.** Let $\mathcal{G}$ be a set of quantum gates. Let Π^{MPQC} be a k-party quantum computation protocol, parameterized by a security parameter n. For any circuit C, set $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players, and adversarial (interactive) algorithm $\mathcal{A}$ that performs all interactions of the players in $I_{\mathcal{A}}$, define $\Pi^{\mathsf{MPQC}}_{C,\mathcal{A}} : R_{\mathcal{A}}^{\mathsf{in}} \sqcup \bigsqcup_{i \notin I_{\mathcal{A}}} R_i^{\mathsf{in}} \to R_{\mathcal{A}}^{\mathsf{out}} \sqcup \bigsqcup_{i \notin I_{\mathcal{A}}} R_i^{\mathsf{out}}$ to be the channel that executes the protocol Π^{MPQC} for circuit C by executing the honest interactions of the players in $[k] \setminus I_{\mathcal{A}}$, and letting $\mathcal{A}$ fulfill the role of the players in $I_{\mathcal{A}}$ (See Figure 4.1, (1)).

For a simulator $\mathcal{S}$ that receives inputs in $R^{\mathsf{in}}_{\mathcal{A}}$, then interacts with the ideal functionalities on all interfaces for players in $I_{\mathcal{A}}$, and then produces output in $R^{\mathsf{out}}_{\mathcal{A}}$, let $\mathfrak{I}^{\mathsf{MPQC}}_{C,\mathcal{S}}$ be the ideal functionality described in Definition 4.2.2, for circuit C, simulator $\mathcal{S}$ for players $i \in I_{\mathcal{A}}$, and honest executions (with $b_i = 0$) for players $i \notin I_{\mathcal{A}}$ (See Figure 4.1, (2)). We say that Π^{MPQC} is a *computationally ε-secure quantum k-party computation protocol with abort,* if for all $I_{\mathcal{A}} \subsetneq [k]$, for all quantum polynomial-time (QPT) adversaries $\mathcal{A}$, and all circuits C comprised of gates from $\mathcal{G}$, there exists a QPT simulator $\mathcal{S}$ such that for all QPT environments $\mathcal{E}$,

$$\left|\Pr\left[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi^{\mathsf{MPQC}}_{C,\mathcal{A}})\right] - \Pr\left[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}^{\mathsf{MPQC}}_{C,\mathcal{S}})\right]\right| \leq \varepsilon. \tag{4.1}$$

Here, the notation $b \leftarrow (\mathcal{E} \leftrightarrows (\cdot))$ represents the environment $\mathcal{E}$, on input 1^n, interacting with the (real or ideal) functionality $(\cdot)$, and producing a single bit b as output.

Remark. In the above definition, we assume that all QPT parties are polynomial in the size of circuit $|C|$, and in the security parameter n.

4.3 Overview of our protocol

The rest of this chapter will be concerned with stating our k-player MPQC protocol, and proving its security. We start by describing some details of the input encoding (see Section 4.4 for full details), and the protocol for circuits consisting only of classically-controlled Clifford operations and measurements (see Section 4.5). Such circuits suffice to perform magic-state computation and distillation, so that the protocol can be extended to arbitrary circuits (see Section 4.6).

The Clifford+measurement protocol consists of several subprotocols, of which we highlight four here: input encoding, public authentication test, single-qubit gate application, and CNOT application. In the following description, the classical MPC is treated as a trusted third party with memory. The general idea is to first ensure that initially all inputs are properly encoded into the Clifford authentication code, and to test the encoding after each computation step that potentially exposes the encoded qubits to an attack. During the protocol, the encryption keys for the Clifford authentication code are only known to the MPC, who stores them in its internal memory.

Input encoding. For an input qubit $|\psi\rangle$ of player i, the MPC hands each player a circuit for a random $(2n+1)$-qubit Clifford group element. Now player i appends $2n$ "trap" qubits initialized in the $|0\rangle$-state, and applies her Clifford. The state is passed around, and all other players apply their Clifford one-by-one, resulting in a

Clifford-encoded qubit $F\left(\left|\psi\right\rangle \otimes \left|0^{2n}\right\rangle\right)$ for which knowledge of the encoding key F is distributed among all players. The final step is our *public authentication test*, which is used in several of the other subprotocols as well. Its goal is to ensure that all players, including player i, have honestly followed the protocol.

The public authentication test (details). Player i, holding the state $F\left(\left|\psi\right\rangle \otimes \left|0^{2n}\right\rangle\right)$, will measure n out of the $2n$ trap qubits, which should all be 0. To enable player i to measure a random subset of n of the trap qubits, the MPC could instruct her to apply $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes \pi) F^\dagger$ to get $E\left(\left|\psi\right\rangle \otimes |0^n\rangle\right) \otimes |r\rangle$, where π is the unitary that permutes the $2n$ trap qubits, E is a random $(n+1)$ qubit Clifford, and $r \in \{0,1\}^n$ is a random string. Then when player i measures the last n trap qubits, if the encoding was correct, she will obtain r and communicate this to the MPC. However, this only guarantees that the remaining traps are correct up to polynomial error.

To get a stronger guarantee, we replace the random permutation with an element from the sufficiently rich yet still efficiently samplable group of invertible transformations over $\mathbb{F}_2^{2n}$, $\mathrm{GL}(2n, \mathbb{F}_2)$. An element $g \in \mathrm{GL}(2n, \mathbb{F}_2)$ may be viewed as a unitary U_g acting on computational-basis states as $U_g |x\rangle = \left|gx\right\rangle$ where $x \in \{0,1\}^{2n}$. In particular, $U_g \left|0^{2n}\right\rangle = \left|0^{2n}\right\rangle$, so if all traps are in the state $|0\rangle$, applying U_g does not change this, whereas for nonzero x, $U_g |x\rangle = \left|x'\right\rangle$ for a *random* $x' \in \{0,1\}^{2n}$. Thus the MPC instructs player i to apply $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes U_g) F^\dagger$ to the state $F\left(\left|\psi\right\rangle \otimes \left|0^{2n}\right\rangle\right)$, then measure the last n qubits and return the result, aborting if it is not r. Crucially, $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes U_g) F^\dagger$ is given as an element of the Clifford group, hiding the structure of the unitary and, more importantly, the values of r and g. So if player i is dishonest and holds a corrupted state, she can only pass the MPC's test by guessing r. If player i correctly returns r, we have the guarantee that the remaining state is a Clifford-authenticated qubit with n traps, $E\left(\left|\psi\right\rangle \otimes |0^n\rangle\right)$, up to exponentially small error.

Single-qubit Clifford gate application. As in the protocol of Dupuis et al. [DNS12], a single-qubit Clifford is applied by simply updating encryption key held by the MPC. If a state is currently encrypted with a Clifford E, decrypting with a "wrong" key $EG^\dagger$ has the effect of applying G to the state. This strategy also works if the application of the Clifford is controlled by a classical bit (even if the classical bit is known only to the MPC). See also Section 3.4.1.

CNOT application. Applying a CNOT gate to two qubits is slightly more complicated: as they are encrypted separately, we cannot just implement the CNOT via a key update like in the case of single qubit Clifford gates. Instead, we bring the two encoded qubits together, and then run a protocol that is similar to input encoding using the $(2n+2)$-qubit register as "input", but using $2n$ additional traps instead of just n, and skipping the final authentication-testing step. The joint state now has $4n+2$ qubits and is encrypted with some Clifford D only known to the MPC. Afterwards, CNOT can be applied via a key update, similarly to single-qubit Cliffords. To split up the qubits again afterwards, the executing player applies $(F_1 \otimes F_2)D^\dagger$, where F_1 and F_2 are freshly sampled by the MPC. The two encoded qubits can then be tested separately using the public authentication test.

As described above, our protocol consists of several subprotocols. To show the security of the full protocol, one may be tempted to define ideal functionalities for each of the subprotocols: an ideal application of a single-qubit gate, for example. This strategy will not work, because such a chain of ideal functionalities is unable to reflect the authentication key that is carried over via the internal memory of the MPC, and correlates the functionality of the different subprotocols.

We solve this difficulty by adopting an inductive strategy to proving security. We define a sequence of ideal functionalities, each one comprising a slightly larger functionality than the last. The first ideal functionality only encodes the inputs; The next ideal functionality applies the first gate of the circuit *and* encodes the result; The next applies the first two gates and encodes the result; Et cetera, until we arrive at an ideal functionality that applies the entire circuit. This approach allows us to focus on analyzing one subprotocol at a time, without losing the key correlation between the different subprotocols.

4.3.1 Pauli filter

In our protocol, we repeatedly use a technique which alters a channel that would act jointly on registers S and T, so that its actions on S are replaced by a flag bit into a separate register. The flag is set to 0 if the actions on S belong to some set $\mathcal{P}$, or to 1 otherwise. This way, the new channel "filters" the allowed actions on S. In this section, we describe this tool.

Definition 4.3.1 (Pauli filter)**.** For registers S and T with $|T| > 0$, let U^{ST} be a unitary, and let $\mathcal{P} \subseteq \left(\{0,1\}^{\log_2 |S|}\right)^2$ contain pairs of bit strings. The $\mathcal{P}$-filter of U on register S, denoted $\mathsf{PauliFilter}^S_{\mathcal{P}}(U)$, is the map $T \to TF$ (where F is some single-qubit flag register) that results from the following operations:

1. Initialize two separate registers S and S' in the state $|\Phi\rangle\langle\Phi|$, where

$$|\Phi\rangle := \left|\Phi^+\right\rangle^{\otimes \log_2 |S|} = \left(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\right)^{\otimes \log_2 |S|} \tag{4.2}$$

   consists of $\log_2 |S|$ EPR pairs. Half of each pair is stored in S, the other in S'.

2. Run U on ST.

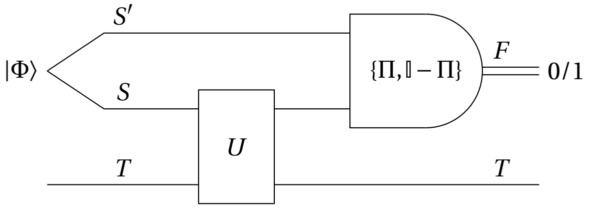


Figure 4.2: The circuit for $\mathsf{PauliFilter}^S_{\mathcal{P}}(U)$ from Definition 4.3.1. By the definition of Π in Equation (4.3), the choice of the set $\mathcal{P}$ determines which types of Paulis are filtered out by the measurement $\{\Pi, \mathbb{I} - \Pi\}$. The $\mathsf{IdFilter}$ and $\mathsf{XFilter}$ are special cases of this filter. Replacing $|\Phi\rangle$ with a different initial state yields a wider array of filters, e.g., the $\mathsf{ZeroFilter}$.

3. Measure SS' with the projective measurement $\{\Pi, \mathbb{I} - \Pi\}$ for

$$\Pi := \sum_{(a,b)\in\mathcal{P}} \left(\mathsf{X}^a\mathsf{Z}^b\right)^S |\Phi\rangle\langle\Phi| \left(\mathsf{Z}^b\mathsf{X}^a\right). \tag{4.3}$$

   If the outcome is Π, set the F register to $|0\rangle\langle 0|$. Otherwise, set it to $|1\rangle\langle 1|$.

Figure 4.2 depicts the circuit for the Pauli filter. Its functionality becomes clear in the following lemma, which we prove by straightforward calculation:

Lemma 4.3.2. *For registers S and T with $|T| > 0$, let U^{ST} be a unitary, and let $\mathcal{P} \subseteq \left(\{0,1\}^{\log_2|S|}\right)^2$. Write $U = \sum_{x,z} (\mathsf{X}^x\mathsf{Z}^z)^S \otimes U^T_{x,z}$. Then running $\mathsf{PauliFilter}^S_{\mathcal{P}}(U)$ on register T equals the map $T \to TF$:*

$$(\cdot)^T \mapsto \sum_{(a,b)\in\mathcal{P}} U^T_{a,b}(\cdot)U^\dagger_{a,b} \otimes |0\rangle\langle 0|^F + \sum_{(a,b)\notin\mathcal{P}} U^T_{a,b}(\cdot)U^\dagger_{a,b} \otimes |1\rangle\langle 1|^F.$$

Proof. Given an arbitrary state ρ^{TR} (for some reference system R), we calculate the result of applying $\mathsf{PauliFilter}^S_{\mathcal{P}}(U)$ to ρ. The state in TR corresponding to $|0\rangle\langle 0|$ in the flag register F is:

$$\mathrm{Tr}_{SS'}\left[\Pi U^{ST}\left(|\Phi\rangle\langle\Phi|^{SS'} \otimes \rho^{TR}\right)U^\dagger\right] \tag{4.4}$$

$$= \sum_{\substack{(a,b)\in\mathcal{P}\\ x,z,x',z'}} \mathrm{Tr}_{SS'}\left[\mathsf{X}^a\mathsf{Z}^b|\Phi\rangle\langle\Phi|\mathsf{Z}^b\mathsf{X}^a\mathsf{X}^x\mathsf{Z}^z|\Phi\rangle\langle\Phi|\mathsf{Z}^{z'}\mathsf{X}^{x'}\right] \otimes U^T_{x,z}\rho^{TR}U^\dagger_{x',z'} \tag{4.5}$$

$$= \sum_{\substack{(a,b)\in\mathcal{P}\\ x,z,x',z'}} \mathrm{Tr}_{SS'}\left[|\Phi\rangle\langle\Phi|\mathsf{X}^{a\oplus x}\mathsf{Z}^{b\oplus z}|\Phi\rangle\langle\Phi|\mathsf{Z}^{b\oplus z'}\mathsf{X}^{a\oplus x'}\right] \otimes U^T_{x,z}\rho^{TR}U^\dagger_{x',z'} \cdot (-1)^{b\cdot(x\oplus x')} \tag{4.6}$$

$$= \sum_{(a,b)\in\mathcal{P}} U^T_{a,b}\rho^{TR}U^\dagger_{a,b}. \tag{4.7}$$

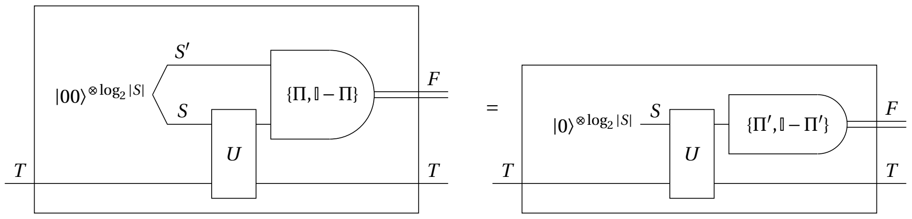

S'
$|00\rangle^{\otimes \log_2 |S|}$
S
$\{\Pi, \mathbb{I} - \Pi\}$
F
U
T
T
$=$
$|0\rangle^{\otimes \log_2 |S|}$
S
$\{\Pi', \mathbb{I} - \Pi'\}$
F
U
T
T

Figure 4.3: The Pauli filter ZeroFilter, where the initial state $|\Phi\rangle$ is replaced by $|00\rangle^{\log_2 |S|}$. This filter, with measurement $\Pi := |00\rangle\langle 00|^{\otimes \log_2 |S|}$, is equivalent to the map which does not prepare the register S', and measures with $\Pi' := |0\rangle\langle 0|^{\otimes \log_2 |S|}$.

The calculation for the $|1\rangle\langle 1|$-flag is very similar, after observing that

$$\mathbb{I} - \sum_{(a,b)\in\mathcal{P}} \mathsf{X}^a\mathsf{Z}^b |\Phi\rangle\langle\Phi| \mathsf{Z}^b\mathsf{X}^a = \sum_{(a,b)\notin\mathcal{P}} \mathsf{X}^a\mathsf{Z}^b |\Phi\rangle\langle\Phi| \mathsf{Z}^b\mathsf{X}^a. \tag{4.8}$$

$\square$

A special case of the Pauli filter for $\mathcal{P} = \{(0^{\log_2 |S|}, 0^{\log_2 |S|})\}$ is due to Broadbent and Wainewright [BW16]. This choice of $\mathcal{P}$ represents only identity: the operation $\mathsf{PauliFilter}_{\mathcal{P}}$ filters out any components of U that do not act as identity on S. We will denote this type of filter with the name IdFilter.

In this work, we will also use $\mathsf{XFilter}^S(U)$, which only accepts components of U that act trivially on register S in the computational basis. It is defined by choosing $\mathcal{P} = \{0^{\log_2 |S|}\} \times \{0,1\}^{\log_2 |S|}$.

Finally, we note that the functionality of the Pauli filter given in Definition 4.3.1 can be generalized, or weakened in a sense, by choosing a different state than $|\Phi\rangle\langle\Phi|$. In this work, we will use the $\mathsf{ZeroFilter}^S(U)$, which initializes SS' in the state $|00\rangle^{\log_2 |S|}$, and measures using the projector $\Pi = |00\rangle\langle 00|$. It filters U by allowing only those Pauli operations that leave the computational-zero state (but not necessarily any other computational-basis states) unaltered:

$$(\cdot) \mapsto U_0^T (\cdot) U_0^\dagger \otimes |0\rangle\langle 0|^F + \sum_{a\neq 0} U_a^T (\cdot) U_a^\dagger \otimes |1\rangle\langle 1|^F, \tag{4.9}$$

where we abbreviate $U_a := \sum_b U_{a,b}$. Note that for $\mathsf{ZeroFilter}^S(U)$, the extra register S' can also be left out (see Figure 4.3).

4.4 Protocol: setup and encoding

4.4.1 Input encoding

In the first phase of the protocol, all players encode their input registers qubit-by-qubit. For simplicity of presentation, we pretend that player 1 holds a single-qubit input state, and the other players do not have input. In the actual protocol, multiple players can hold multiple-qubit inputs: in that case, the initialization is run several times in parallel, using independent randomness. Any other player i can trivially take on the role of player 1 by relabeling the player indices.

Definition 4.4.1 (Ideal functionality for input encoding)**.** Without loss of generality, let R_1^{in} be a single-qubit input register, and let $\dim(R_i^{\mathsf{in}}) = 0$ for all $i \neq 1$. Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.

1. Player 1 sends register R_1^{in} to the trusted third party.
2. The trusted third party initializes a register T_1 with $|0^n\rangle\langle 0^n|$, applies a random $(n+1)$-qubit Clifford E to MT_1, and sends these registers to player 1.
3. All players $i \in I_{\mathcal{A}}$ send a bit b_i to the trusted third party. If $b_i = 0$ for all i, then the trusted third party stores the key E in the state register S of the ideal functionality. Otherwise, it aborts by storing $\perp$ in S.

The following protocol implements the ideal functionality. It uses, as a black box, an ideal functionality MPC that implements a classical multi-party computation with memory.

Protocol 4.4.2. (Input encoding) Without loss of generality, let $M := R_1^{\mathsf{in}}$ be a single-qubit input register, and let $|R_i^{\mathsf{in}}| = 0$ for all $i \neq 1$.

1. For every $i \in [k]$, MPC samples a random $(2n+1)$-qubit Clifford F_i and tells it to player i.
2. Player 1 applies the map $\rho^M \mapsto F_1\left(\rho^M \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_1T_2}\right)F_1^\dagger$ for two n-qubit (trap) registers T_1 and T_2, and sends the registers MT_1T_2 to player 2.
3. Every player $i = 2,3,\ldots,k$ applies F_i to MT_1T_2, and forwards it to player $i+1$. Eventually, player k sends the registers back to player 1.
4. MPC samples a random $(n+1)$-qubit Clifford E, random n-bit strings r and s, and a random classical invertible linear operator $g \in GL(2n, \mathbb{F}_2)$. Let U_g be the (Clifford) unitary that computes g in-place, i.e., $U_g|t\rangle = \left|g(t)\right\rangle$ for all $t \in \{0,1\}^{2n}$.

5. MPC gives[a]

$$V := \left(E^{MT_1} \otimes (\mathsf{X}^r \mathsf{Z}^s)^{T_2}\right)\left(\mathbb{I} \otimes (U_g)^{T_1 T_2}\right)(F_k \cdots F_2 F_1)^\dagger$$

to player 1, who applies it to MT_1T_2.

6. Player 1 measures T_2 in the computational basis, discarding the measured wires, and keeps the other $(n+1)$ qubits as its output in $R_1^{\mathsf{out}} = MT_1$.

7. Player 1 submits the measurement outcome r' to MPC, who checks whether $r = r'$. If so, MPC stores the key E in its memory-state register S. If not, it aborts by storing $\perp$ in S.

[a] MPC gives V as a group element, and the adversary cannot decompose it into the different parts that appear in its definition.

If MPC aborts the protocol in Step 7, the information about the Clifford encoding key E is erased. In that case, the registers MT_1 will be fully mixed. Note that this result differs slightly from the "reject" outcome of a quantum authentication code as in Definition 3.2.1, where the message register M is replaced by a dummy state $|\perp\rangle\langle\perp|$. In our current setting, the register M is in the hands of (the possibly malicious) player 1. We therefore cannot enforce the replacement of register M with a dummy state: we can only make sure that all its information content is removed. Depending on the application or setting, the trusted MPC can of course broadcast the fact that they aborted to all players, including to the honest one(s).

To run Protocol 4.4.2 in parallel for multiple input qubits held by multiple players, MPC samples a list of Cliffords $F_{i,q}$ for each player $i \in [k]$ and each qubit q. The $F_{i,q}$ operations can be applied in parallel for all qubits q: with k rounds of communication, all qubits will have completed their round past all players.

We will show that Protocol 4.4.2 fulfills the ideal functionality for input encoding:

Lemma 4.4.3. *Let Π^{Enc} be Protocol 4.4.2, and $\mathfrak{I}^{\mathsf{Enc}}$ be the ideal functionality described in Definition 4.4.1. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{\mathsf{Enc}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{\mathsf{Enc}})| \leq \mathrm{negl}(n).$$

Note that the environment $\mathcal{E}$ also receives the state register S, which acts as the "output" register of the ideal functionality (in the simulated case) or of MPC (in the real case). It is important that the environment cannot distinguish between the output states even given that state register S, because we want to be able to compose Protocol 4.5.5 with other protocols that use the key information inside S. In other words, it is important that, unless the key is discarded, the plaintext states *inside* the Clifford encoding are also indistinguishable for the environment.

We start by providing a sketch of the proof for Lemma 4.4.3, and then state and prove an auxiliary lemma before its full proof.

Proof sketch. We divide our proof into two cases: when player 1 is honest, or when she is dishonest.

For the case when player 1 is honest, we know that she correctly prepares the expected state before the state is given to the other players. That is, she appends $2n$ ancilla qubits in state $|0\rangle$ and applies the random Clifford instructed by the classical MPC. When the encoded state is returned to player 1, she performs the Clifford V as instructed by the MPC. By the properties of the Clifford encoding, if the other players acted dishonestly, the tested traps will be nonzero with probability exponentially close to 1.

The second case is a bit more complicated: the first player has full control over the state and, more importantly, the traps that will be used in the first encoding. In particular, she could start with nonzero traps, which could possibly give some advantage to the dishonest players later on the execution of the protocol.

In order to prevent this type of attack, the MPC instructs the first player to apply a random linear function U_g on the traps, which is hidden from the players inside the Clifford V. If the traps were initially zero, their value does not change, but otherwise, they will be mapped to a random value, unknown by the dishonest parties. As such, the map U_g removes any advantage that the dishonest parties could have in Step 7 by starting with nonzero traps. Because *any* nonzero trap state in T_1T_2 is mapped to a random string, it suffices to measure only T_2 in order to be convinced that T_1 is also in the all-zero state (except with negligible probability). This intuition is formalized in Lemma 4.4.4 below.

Other possible attacks are dealt with in a way that is similar to the case where player 1 is honest (but from the perspective of another honest player).

In the full proof, we present two simulators, one for each case, that tests (using Pauli filters from Section 4.3.1) whether the adversary performs any such attacks during the protocol, and chooses the input to the ideal functionality accordingly. See Figure 4.4 for a pictorial representation of the structure of the simulator for the case where player 1 is honest. □

Before we prove Lemma 4.4.3, let us begin by zooming in on the test phase (Steps 4–6): we show in a separate lemma that, with high probability, it only checks out if the resulting state is a correctly-encoded one.

For a projector Π on two n-qubit quantum registers T_1 and T_2, define the quantum channel $\mathcal{L}_\Pi$ on T_1T_2 by

$$\mathcal{L}_\Pi(X) := \Pi(X)\Pi + |\bot\rangle\langle\bot| \operatorname{Tr}\left[\bar{\Pi}X\right] \tag{4.10}$$

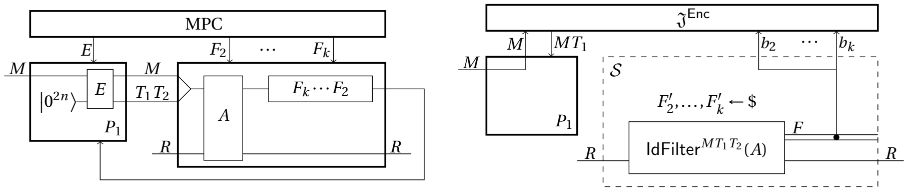


Figure 4.4: On the left, the adversary's interaction with the protocol Π^{Enc} in case player 1 is the only honest player. The R register contains side information for the adversary. We may assume that the adversarial map consists of a unitary A followed by the honest protocol $F_k \cdots F_2$ (see the full proof of Lemma 4.4.3). On the right, the simulator's interaction with $\mathfrak{J}^{\mathsf{Enc}}$. It performs the Pauli filter $\mathsf{IdFilter}^{MT_1T_2}$ on the adversary's attack on the encoded state.

where $|\bot\rangle$ is a distinguished state on T_1T_2 with $\Pi|\bot\rangle = 0$.

Furthermore, for $s \in \{0,1\}^n$, define the "full" and "half" projectors

$$\Pi_{s,F} := \begin{cases} |0^{2n}\rangle\langle 0^{2n}|^{T_1T_2} & \text{if } s = 0^n \\ 0 & \text{else} \end{cases} \tag{4.11}$$

$$\Pi_{s,H} := \mathbb{I}^{T_1} \otimes |s\rangle\langle s|^{T_2}. \tag{4.12}$$

The following lemma shows that the full measurement $\Pi_{s,F}$ is an equivalent test to applying a twirl $\mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}$ followed by a half measurement of $\Pi_{s,H}$.

Lemma 4.4.4. *For any $s \in \{0,1\}^n$, applying a random element of* $\mathrm{GL}(2n,\mathbb{F}_2)$ *followed by $\mathcal{L}_{\Pi_{s,H}}$ (see Equation* (4.10)*) is essentially equivalent to applying $\mathcal{L}_{\Pi_{s,F}}$:*

$$\left\| \mathcal{L}_{\Pi_{s,F}} - \mathcal{L}_{\Pi_{s,H}} \circ \mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)} \right\|_\diamond \leqslant 8 \cdot 2^{-\frac{n}{2}} = \mathrm{negl}(n).$$

Proof. First, observe the following facts about a random $g \in \mathrm{GL}(2n,\mathbb{F}_2)$. Of course, $g0 = 0$ by linearity. On the other hand, gx is uniformly random on $\mathbb{F}_2^{2n} \setminus \{0\}$ for $x \neq 0$. More generally, nonzero x, y are linearly independent if and only if $x \neq y$: for those values of x and y, (gx, gy) is uniformly random on $\left\{(x', y') \in \left(\mathbb{F}_2^{2n} \setminus \{0\}\right)^2 \,|\, x' \neq y'\right\}$.

We want to calculate the effect of twirling over $\mathrm{GL}(2n,\mathbb{F}_2)$, followed by the half projector, for an arbitrary state $\rho^{T_1T_2E}$. Throughout this proof, we will abbreviate $\mathcal{T} := \mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}$. Expanding ρ as $\rho = \sum_{x,y\in\{0,1\}^{2n}} |x\rangle\langle y|^{T_1T_2} \otimes \langle x|\rho|y\rangle^E$, we have

$$\mathcal{L}_{\Pi_{s,H}}^{T_1T_2}\left(\mathcal{T}^{T_1T_2}(\rho)\right) = \sum_{x,y\in\{0,1\}^{2n}} \mathcal{L}_{\Pi_{s,H}}^{T_1T_2}\left(\mathcal{T}\left(|x\rangle\langle y|\right)\right) \otimes \langle x|\rho|y\rangle. \tag{4.13}$$

Thus, it suffices to analyze the effect of the twirl and projector on states of the form $|x\rangle\langle y|$. Define the unit vector

$$|+'\rangle := \left(2^{2n} - 1\right)^{-\frac{1}{2}} \sum_{x\in\mathbb{F}_2^{2n}\setminus\{0\}} |x\rangle \tag{4.14}$$

as the superposition over all nonzero elements x. We calculate, for $x, y \in \mathbb{F}_2^{2n} \setminus \{0\}$ with $x \neq y$,

$$\mathcal{T}(|0\rangle\langle 0|) = |0\rangle\langle 0|; \tag{4.15}$$

$$\mathcal{T}(|x\rangle\langle x|) = \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1}; \tag{4.16}$$

$$\begin{aligned}\mathcal{T}(|x\rangle\langle 0|) &= \left(2^{2n}-1\right)^{-1} \sum_{x \in \mathbb{F}_2^{2n} \setminus \{0\}} |x\rangle\langle 0| \\ &= \left(2^{2n}-1\right)^{-\frac{1}{2}} \left|+'\right\rangle\!\left\langle 0\right|;\end{aligned} \tag{4.17}$$

$$\begin{aligned}\mathcal{T}(\left|x\right\rangle\!\left\langle y\right|) &= \left(2^{2n}-1\right)^{-1} \left(2^{2n}-2\right)^{-1} \sum_{\substack{z,t \in \mathbb{F}_2^{2n} \setminus \{0\} \\ z \neq t}} |z\rangle\langle t| \\ &= \left(2^{2n}-2\right)^{-1} \left(\left|+'\right\rangle\!\left\langle +'\right| - \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1}\right) =: S.\end{aligned} \tag{4.18}$$

Then, in order to analyze the effect of the half projector, we first calculate the subexpressions

$$\left\| \Pi_{s,H} \left|+'\right\rangle \right\|_2 \leqslant \sqrt{\frac{2^n}{2^{2n}-1}}; \tag{4.19}$$

$$\left\| \Pi_{s,H} \left|+'\right\rangle\!\left\langle 0\right| \Pi_{s,H} \right\|_{\mathrm{tr}} \leqslant \sqrt{\frac{2^n}{2^{2n}-1}}; \tag{4.20}$$

$$\left\| \Pi_{s,H} \left|+'\right\rangle\!\left\langle +'\right| \Pi_{s,H} \right\|_{\mathrm{tr}} \leqslant \frac{2^n}{2^{2n}-1}; \tag{4.21}$$

$$\left\| \Pi_{s,H} \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1} \Pi_{s,H} \right\|_{\mathrm{tr}} \leqslant \frac{2^n}{2^{2n}-1}. \tag{4.22}$$

Equations (4.20) and (4.21) follow from Equation (4.19) by Hölder's inequality (see Section 2.4.3).

We use the above inequalities, and the fact that $\| \rho \|_{\mathrm{tr}} = 1$, to compute the effect of the GL$(2n, \mathbb{F}_2)$-twirl and the half projector on states of the form $|0\rangle\langle 0|$, $|x\rangle\langle x|$, $|x\rangle\langle 0|$, and $\left|x\right\rangle\!\left\langle y\right|$, for $x \neq 0 \neq y$ and $x \neq y$. For the state $|0\rangle\langle 0|$, we are interested to know the exact output state:

$$\Pi_{s,H} \mathcal{T}(|0\rangle\langle 0|) \Pi_{s,H} \otimes \langle 0| \rho |0\rangle = \begin{cases} |0\rangle\langle 0|^{T_1 T_2} \otimes \langle 0| \rho |0\rangle & \text{if } s = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.23}$$

For the other three cases, we are merely interested in their trace norm: we will show that this norm is, in all cases, negligible in n. For states of the form $|x\rangle\langle x|$ with $x \neq 0$, we use triangle inequality of the trace norm to obtain

$$\left\| \sum_{x \neq 0} \Pi_{s,H} \mathcal{T}(|x\rangle\langle x|) \Pi_{s,H} \otimes \langle x| \rho |x\rangle \right\|_{\mathrm{tr}} \tag{4.24}$$

$$\overset{(4.16)}{=} \left\| \Pi_{s,H} \frac{\mathbb{1} - |0\rangle\langle 0|}{2^{2n}-1} \Pi_{s,H} \otimes \sum_{x\neq 0} \langle x|\rho|x\rangle \right\|_{\mathrm{tr}} \tag{4.25}$$

$$= \left\| \Pi_{s,H} \frac{\mathbb{1} - |0\rangle\langle 0|}{2^{2n}-1} \Pi_{s,H} \right\|_{\mathrm{tr}} \cdot \left\| \mathrm{Tr}_{T_1 T_2} \left[(\mathbb{1} - |0\rangle\langle 0|)\, \rho \right] \right\|_{\mathrm{tr}} \tag{4.26}$$

$$\overset{(4.22),\ \text{Lem. } 2.4.3}{\leqslant} \frac{2^n}{2^{2n}-1} \cdot \| \mathbb{1} - |0\rangle\langle 0| \|_{\infty} \cdot \| \rho \|_{\mathrm{tr}} \tag{4.27}$$

$$\leqslant \frac{2^n}{2^{2n}-1}. \tag{4.28}$$

For states of the form $|x\rangle\langle 0|$ (and, analogously, those of the form $|0\rangle\langle x|$) with $x \neq 0$, a similar derivation gives

$$\left\| \sum_{x\neq 0} \Pi_{s,H} \mathcal{T}(|x\rangle\langle 0|)\, \Pi_{s,H} \otimes \langle x|\rho|0\rangle \right\|_{\mathrm{tr}} \tag{4.29}$$

$$\overset{(4.17)}{=} \left\| \left(2^{2n}-1\right)^{-\frac{1}{2}} \Pi_{s,H} \left|+'\right\rangle\!\left\langle 0\right| \Pi_{s,H} \otimes \sum_{x\neq 0} \langle x|\rho|0\rangle \right\|_{\mathrm{tr}} \tag{4.30}$$

$$= \left\| \Pi_{s,H} \left|+'\right\rangle\!\left\langle 0\right| \Pi_{s,H} \right\|_{\mathrm{tr}} \cdot \left\| \left\langle +'\right| \rho |0\rangle \right\|_{\mathrm{tr}} \tag{4.31}$$

$$\overset{(4.20),\ \text{Lem. } 2.4.3}{\leqslant} \sqrt{\frac{2^n}{2^{2n}-1}} \cdot \left\| \left|0\right\rangle\!\left\langle +'\right| \right\|_{\infty} \cdot \| \rho \|_{\mathrm{tr}} \tag{4.32}$$

$$\leqslant \sqrt{\frac{2^n}{2^{2n}-1}}. \tag{4.33}$$

And finally, for states of the form $\left|x\right\rangle\!\left\langle y\right|$ with $x \neq 0 \neq y$ and $x \neq y$, we use the triangle inequality of the trace norm to derive

$$\left\| \sum_{\substack{x\neq 0\neq y \\ x\neq y}} \Pi_{s,H} \mathcal{T}\!\left(\left|x\right\rangle\!\left\langle y\right|\right) \Pi_{s,H} \otimes \langle x|\rho\left|y\right\rangle \right\|_{\mathrm{tr}} \tag{4.34}$$

$$\overset{(4.18)}{=} \left\| \Pi_{s,H} S \Pi_{s,H} \otimes \sum_{\substack{x\neq 0\neq y \\ x\neq y}} \langle x|\rho\left|y\right\rangle \right\|_{\mathrm{tr}} \tag{4.35}$$

$$= \left\| \Pi_{s,H} S \Pi_{s,H} \otimes \mathrm{Tr}_{T_1 T_2} \left[\left(\left(2^{2n}-1\right) \left|+'\right\rangle\!\left\langle +'\right| - (\mathbb{1} - |0\rangle\langle 0|) \right) \rho \right] \right\|_{\mathrm{tr}} \tag{4.36}$$

$$\leqslant \left(2^{2n}-1\right) \left\| \Pi_{s,H} S \Pi_{s,H} \otimes \mathrm{Tr}_{T_1 T_2} \left[\left|+'\right\rangle\!\left\langle +'\right| \rho \right] \right\|_{\mathrm{tr}} + \left\| \Pi_{s,H} S \Pi_{s,H} \otimes \mathrm{Tr}_{T_1 T_2} \left[(\mathbb{1} - |0\rangle\langle 0|)\, \rho \right] \right\|_{\mathrm{tr}}. \tag{4.37}$$

We continue the derivation similarly to the previous derivations, using the multiplicativity of the trace norm w.r.t. the Kronecker product, combined with Lemma 2.4.3 and several inequalities established above:

$$
\begin{aligned}
\text{Eq. (4.37)} &\overset{\text{Lem. (2.4.3)}}{\leqslant} \left(2^{2n}-1\right) \left\| \Pi_{s,H} S \Pi_{s,H} \right\|_{\mathrm{tr}} \cdot \left\| |+'\rangle\langle +'| \right\|_{\infty} \cdot \left\| \rho \right\|_{\mathrm{tr}} \\
&\qquad + \left\| \Pi_{s,H} S \Pi_{s,H} \right\|_{\mathrm{tr}} \cdot \left\| \mathbb{I} - |0\rangle\langle 0| \right\|_{\infty} \cdot \left\| \rho \right\|_{\mathrm{tr}} && (4.38) \\
&\leqslant \quad 2^{2n} \left\| \Pi_{s,H} S \Pi_{s,H} \right\|_{\mathrm{tr}} && (4.39) \\
&\overset{(4.18)}{\leqslant} \quad \frac{2^{2n}}{2^{2n}-2} \left(\left\| \Pi_{s,H} |+'\rangle\langle +'| \Pi_{s,H} \right\|_{\mathrm{tr}} + \left\| \Pi_{s,H} \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1} \Pi_{s,H} \right\|_{\mathrm{tr}} \right) && (4.40) \\
&\overset{(4.21),(4.22)}{\leqslant} \frac{2^{2n}}{2^{2n}-2} \cdot \left(\frac{2^n}{2^{2n}-1} + \frac{2^n}{2^{2n}-1} \right) && (4.41) \\
&\leqslant \quad 4 \cdot \frac{2^n}{2^{2n}-1}. && (4.42)
\end{aligned}
$$

Using the inequalities obtained above, let us now compare the effect of the full projector,

$$
\Pi_{s,F}^{T_1T_2} \rho^{T_1T_2E} \Pi_{s,F} = \begin{cases} |0\rangle\langle 0|^{T_1T_2} \otimes \left(\langle 0| \rho |0\rangle\right)^E & \text{if } s = 0 \\ 0 & \text{otherwise,} \end{cases} \tag{4.43}
$$

to the effect of the twirl followed by the half projector. The term for $|0\rangle\langle 0|$ will cancel, and by the triangle inequality, we have

$$
\begin{aligned}
\left\| \mathcal{L}_{\Pi_{s,F}}(\rho) - \mathcal{L}_{\Pi_{s,H}}\left(\mathcal{T}(\rho)\right) \right\|_{\mathrm{tr}} &\leqslant \left\| \sum_{x \neq 0} \Pi_{s,H} \mathcal{T}(|x\rangle\langle x|) \Pi_{s,H} \otimes \langle x| \rho |x\rangle \right\|_{\mathrm{tr}} \\
&\quad + \left\| \sum_{x \neq 0} \Pi_{s,H} \mathcal{T}(|x\rangle\langle 0|) \Pi_{s,H} \otimes \langle x| \rho |0\rangle \right\|_{\mathrm{tr}} \\
&\quad + \left\| \sum_{x \neq 0} \Pi_{s,H} \mathcal{T}(|0\rangle\langle x|) \Pi_{s,H} \otimes \langle 0| \rho |x\rangle \right\|_{\mathrm{tr}} \\
&\quad + \left\| \sum_{\substack{x \neq 0 \neq y \\ x \neq y}} \Pi_{s,H} \mathcal{T}(|x\rangle\langle y|) \Pi_{s,H} \otimes \langle x| \rho |y\rangle \right\|_{\mathrm{tr}} && (4.44) \\
&\leqslant 2 \cdot \sqrt{\frac{2^n}{2^{2n}-1}} + \frac{2^n}{2^{2n}-1} + 4 \cdot \frac{2^n}{2^{2n}-1} && (4.45) \\
&\leqslant 8 \cdot 2^{-\frac{n}{2}}, && (4.46)
\end{aligned}
$$

which is negligible. Since the bound holds for arbitrary $\rho^{T_1T_2E}$, the bound on the diamond norm follows. □

Now that we have established that it suffices to measure only the T_2 register (after applying a random $g \in GL(2n, \mathbb{F}_2)$), we are ready to prove the security of Protocol 4.4.2:

Proof of Lemma 4.4.3. We consider two cases: either player 1 is honest, or she is corrupted.

Case 1: player 1 is honest. For the setting where player 1 is honest, we prove security in the worst case, where *all* other players are corrupted: $I_{\mathcal{A}} = \{2, 3, \ldots, k\}$. If, instead, some of these players are not corrupted, a simulator can simulate the actions of every honest player $h \neq 1$ (by applying a random Clifford), and interleave these honest actions with the adversarial maps of the corrupted players. The resulting map is a special case of the adversarial map we consider below. Since the only task of the honest players $h \neq 1$ is to apply a random Clifford, it is sufficient if the simulator samples this Clifford itself.

The corrupted players act as one entity whose honest action is to apply $U_H := F_k F_{k-1} \cdots F_3 F_2$, and return the state to player 1 (see the left-hand side of Figure 4.4.). Without loss of generality, assume that $\mathcal{A}$ is unitary by expanding the side-information register R as necessary. Then, define an attack unitary $A := U_H^\dagger \mathcal{A}$, so that we may write $\mathcal{A} = U_H A$. In other words, we establish that $\mathcal{A}$ consists of a unitary attack A, followed by the honest unitary U_H. Note that A may depend arbitrarily on its instructions F_2 through F_k.

The simulator $\mathcal{S}$ has access to the ideal functionality only through the ability to submit the bits b_i for players $i \neq 1$. It does not receive any input from the environment, except for a side-information register R. Define the simulator as follows (in terms of an adversarial map A):

Simulator 4.4.5 (see the right-hand side of Figure 4.4)**.** On input register R received from the environment, do:

1. Sample random $F_2', \ldots, F_k' \in \mathscr{C}_{2n+1}$.[a]
2. Run $\mathsf{IdFilter}^{MT_1T_2}(A)$ on the register R, using the instructions $F_2', F_3', \ldots, F_k'$ to determine A. (See Section 4.3.1.)
3. If the flag register is 0, set $b_i = 0$ for all $i \neq 1$. Otherwise, set $b_i = 1$ for all $i \neq 1$. Submit the bits b_i to the ideal functionality.

[a]Whenever a simulator samples random elements, it does so by running the ideal functionality for classical MPC with the adversary it is currently simulating. If that ideal functionality aborts, the simulator will also abort by setting $b_i = 1$ for the adversarial players i. In that case, the simulated output state and the real output state will be indistinguishable by security of the classical MPC. To avoid clutter in the exposition of our simulators and proofs, we will ignore this technicality, and pretend that the simulator generates the randomness itself.

We will consider the joint state in the output register $R_1^{\mathsf{out}} = MT_1$, the state register S, and the attacker's side-information register R in both the real and the ideal (simulated) case. In both cases, it will be useful to decompose the attack map A as

$$A = \sum_{a,c \in \{0,1\}^{n+1}} \left(\mathsf{X}^a \mathsf{Z}^c\right)^{MT_1} \otimes A^R_{a,c}. \tag{4.47}$$

We start by analyzing the ideal case. By Definition 4.4.1 of the ideal encoding functionality and Lemma 4.3.2, and using $\mathcal{P} = \{(0,0)\}$ with 0 as an abbreviation for 0^n, the output state in MT_1RS in case of accept (setting all $b_i = 0$) is

$$\mathop{\mathbb{E}}_{E} E^{MT_1} \left(A^R_{0,0} \rho^{MR} A^\dagger_{0,0} \otimes |0^n\rangle\langle 0^n|^{T_1}\right) E^\dagger \otimes |E\rangle\langle E|^S. \tag{4.48}$$

The output state in MT_1RS in case of reject is (by Definition 4.4.1 and Lemma 4.3.2)

$$\mathcal{T}^{MT_1}_{\mathscr{C}_{n+1}} \left(\sum_{(x,z)\neq(0,0)} A^R_{x,z} \rho^{MR} A^\dagger_{x,z} \otimes |0^n\rangle\langle 0^n|^{T_1}\right) \otimes |\bot\rangle\langle\bot|^S \tag{4.49}$$

$$= \tau^{MT_1} \otimes \sum_{(x,z)\neq(0,0)} A_{x,z} \rho_R A^\dagger_{x,z} \otimes |\bot\rangle\langle\bot|^S. \tag{4.50}$$

Next, we consider the state in MT_1RS after the real protocol is executed, and argue that it is negligibly close to Equations (4.48)+(4.50). Again, we first consider the accept case. Following the steps in Protocol 4.4.2 on an input state ρ^{MR}, and noting that

$$\left(F_1^\dagger\right)^{MT_1T_2} A^{MT_1T_2R} F_1 \left(\rho^{MR} \otimes |0^{2n}\rangle\langle 0^{2n}|^{T_1T_2}\right) F_1^\dagger A^\dagger F_1 = \left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right), \tag{4.51}$$

the output state in the case of accept is

$$\mathop{\mathbb{E}}_{E,r,s} \langle r|^{T_2} \left(E \otimes \mathsf{X}^r \mathsf{Z}^s\right) \mathcal{T}^{T_1T_2}_{\mathrm{GL}(2n,\mathbb{F}_2)} \left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right)\right) \left(E \otimes \mathsf{X}^r \mathsf{Z}^s\right)^\dagger |r\rangle \otimes |E\rangle\langle E|^S \tag{4.52}$$

$$= \mathop{\mathbb{E}}_{E} E^{MT_1} \langle 0^n|^{T_2} \mathcal{T}^{T_1T_2}_{\mathrm{GL}(2n,\mathbb{F}_2)} \left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right)\right) |0^n\rangle E^\dagger \otimes |E\rangle\langle E|^S \tag{4.53}$$

$$\approx_{\mathrm{negl}(n)} \mathop{\mathbb{E}}_{E} E^{MT_1} \langle 0^{2n}|^{T_1T_2} \left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right)\right) |0^{2n}\rangle E^\dagger \otimes |E\rangle\langle E|^S, \tag{4.54}$$

where the approximation follows from Lemma 4.4.4. This is where the authentication property of the Clifford code comes in: by Lemma 3.3.2, only the part of A that acts trivially on MT_1T_2 remains after the measurement of T_1T_2. Thus, Eq. (4.54) $\approx_{\mathrm{negl}(n)}$ Eq. (4.48).

The reject case of the real protocol is similar: again using Lemmas 3.3.2 and 4.4.4, we can see that it approximates (up to a negligible factor in n) Eq. (4.50).

We conclude that, from the point of view of any environment, the real output state in registers MT_1SR (encoding, memory state, and side information) is indistinguishable from the simulated state.

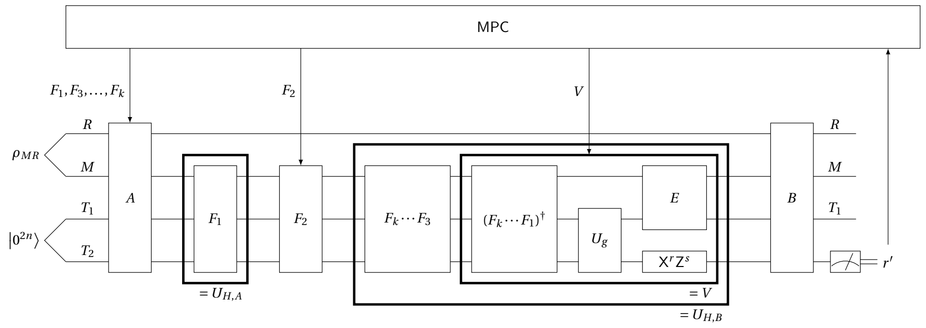


Figure 4.5: Execution of the input-encoding protocol Π^{Enc} (see Protocol 4.4.2), where player 2 is the only honest player (case 2).

Case 2: player 1 is dishonest. For the same reason as in the first case, we assume that the only honest player is player 2, i.e., $I_{\mathcal{A}} = \{1, 3, 4, \ldots, k\}$.

In the real protocol, the adversary interacts with the honest player 2, and has two opportunities to attack: before player 2 applies its Clifford operation, and after.

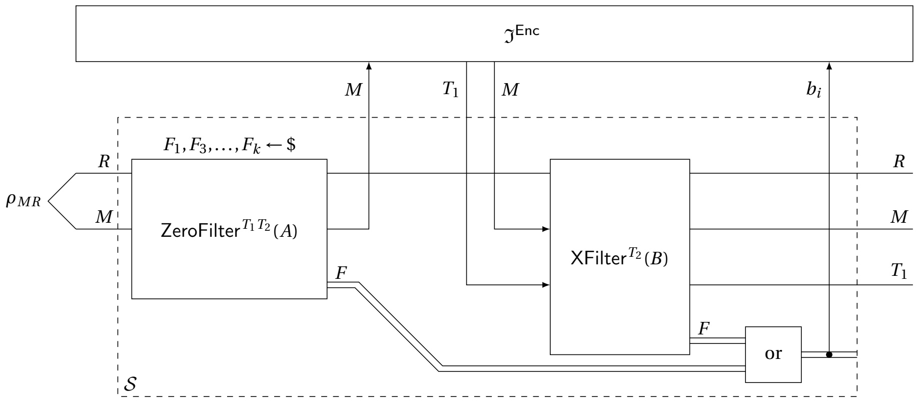
The adversaries' actions before the interaction with player 2 can, without loss of generality, be described by a unitary $U_{H,A} \cdot A$, that acts on the input state ρ^{MR}, plus the registers $T_1 T_2$ that are initialized to zero. The unitary $U_{H,A}$ is player 1's honest operation $F_1^{MT_1T_2}$.

Similarly, the adversaries' actions after the interaction with player 2 can be described by a unitary $B \cdot U_{H,B}$, followed by a computational-basis measurement on T_2 which results in an n-bit string r'. Again, $U_{H,B}$ is the honest unitary $VF_k \cdots F_4 F_3$ that should be applied jointly by players $3, 4, \ldots, k, 1$. See Figure 4.5.

For any adversary, described by such unitaries A and B, define a simulator as follows (see Figure 4.6):

Simulator 4.4.6. On input register MR received from the environment, do:

1. Initialize $b_i = 0$ for all $i \in I_{\mathcal{A}}$.
2. Sample random $F_1, F_2, \ldots, F_k \in \mathscr{C}_{2n+1}$. Run $\mathsf{ZeroFilter}^{T_1T_2}(A)$ on MR, using the instructions F_i (for all $i \in I_{\mathcal{A}}$) to determine A. If the filter flag is 1, abort by setting $b_1 = 1$.
3. Input the M register into the ideal functionality, and receive a state in the register MT_1.



$\mathfrak{I}^{\mathsf{Enc}}$
M
T_1
M
b_i
$F_1, F_3, \ldots, F_k \leftarrow \$$
R
ρ_{MR}
M
$\mathsf{ZeroFilter}^{T_1T_2}(A)$
F
$\mathsf{XFilter}^{T_2}(B)$
F
or
R
M
T_1
$\mathcal{S}$

Figure 4.6: Interaction between the ideal functionality and the simulator $\mathcal{S}$ (see Simulator 4.4.6) for the case in which only player 2 is honest (case 2). The simulator performs two filters, and sets the abort bit to 1 whenever at least one of the flags F is set to 1.

4. Run $\mathsf{XFilter}^{T_2}(B)$ on MT_1R, using the instruction $V := F_3^\dagger F_4^\dagger \cdots F_k^\dagger$ to determine B. (This choice of V ensures that the honest action $U_{H,B}$ is identity.) If the filter flag is 1, abort by setting $b_1 = 1$.

5. Submit all the b_i to the ideal functionality.

Similarly to the previous case, we consider the output state in the registers MT_1RS in both the ideal (simulated) case, and the real case, as computed on an input state ρ^{MR}.

Again, we decompose the attack maps A and B as

$$A = \sum_{a,c\in\{0,1\}^{2n}} \left(\mathsf{X}^a\mathsf{Z}^c\right)^{T_1T_2} \otimes A_{a,c}^{MR} \quad , \tag{4.55}$$

$$B = \sum_{b,d\in\{0,1\}^{n}} \left(\mathsf{X}^b\mathsf{Z}^d\right)^{T_2} \otimes B_{b,d}^{MT_1R} \quad . \tag{4.56}$$

Note that the decompositions are taken over different registers for A and B. In the derivations below, we will often abbreviate $A_a := \sum_c A_{a,c}$, and, in the subscripts, we will abbreviate 0 for the all-zero string.

In the ideal or simulated case, one of three things may happen: the simulator sets b_1 to 0 (signaling accept to the ideal functionality), or sets b_1 to 1 in Step 2, or sets b_1 to 1 in Step 4 (both signaling reject to the ideal functionality). The ideal output state is thus the sum of three separate terms, which we will analyze separately.

We start with the the accept case, where both filters result in a $|0\rangle\langle 0|$ flag. Using the decompositions from Equations (4.55) and (4.56), we apply Lemma 4.3.2 to see that the resulting state is

$$\mathop{\mathbb{E}}_{E}\sum_{d} B_{0,d}^{MT_1R} E\left(A_0 \rho A_0^\dagger \otimes |0^n\rangle\langle 0^n|^{T_1}\right) E^\dagger B_{0,d}^\dagger \otimes |E\rangle\langle E|^S . \tag{4.57}$$

Here, E is the key that the ideal functionality samples (and stores in the register S) when it is called to encode M.

Next, we consider the simulator choosing $b_1 = 1$ already in Step 2, the zero filter has failed. In this case, the ideal functionality does not store the encoding key E in the register S. This allows us to view the Clifford encoding as a twirl on the Clifford group. The output state is (by Lemma 4.3.2)

$$\begin{aligned} &\sum_{a\neq 0^{2n},b,d} B_{b,d}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(A_a\rho^{MR}A_a^\dagger \otimes |0^n\rangle\langle 0^n|^{T_1}\right)B_{b,d}^\dagger \otimes |\bot\rangle\langle\bot|^S & (4.58)\\ =&\sum_{a\neq 0^{2n},b,d} B_{b,d}\left(\mathrm{Tr}_M\left[A_a\rho^{MR}A_a^\dagger\right]\otimes \tau^{MT_1}\right)B_{b,d}^\dagger \otimes |\bot\rangle\langle\bot|^S . & (4.59)\end{aligned}$$

Note that in this case, the flag in the X filter does not influence the bit b_1 (it is already set to 1). Therefore, both terms in Lemma 4.3.2 survive, and all pairs (b, d) are included in the sum.

Finally, we look at the case where the zero filter does not result in changing b_1, but the X filter does, in Step 4. If this happens, the key E is erased so we can again apply a Clifford twirl, and the output state is (by Lemma 4.3.2)

$$\begin{aligned} &\sum_{b\neq 0^n,d} B_{b,d}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(A_0\rho^{MR}A_0^\dagger \otimes |0^n\rangle\langle 0^n|^{T_1}\right)B_{b,d}^\dagger \otimes |\bot\rangle\langle\bot|^S \\ =&\sum_{b\neq 0^n} B_{b,d}\left(\mathrm{Tr}_M\left[A_0\rho^{MR}A_0^\dagger\right]\otimes \tau^{MT_1}\right)B_{b,d}^\dagger \otimes |\bot\rangle\langle\bot|^S . & (4.60)\end{aligned}$$

In summary, the output state in the ideal case is

$$\text{Eq. (4.57)} + \text{Eq. (4.59)} + \text{Eq. (4.60)}. \tag{4.61}$$

In the real protocol, only one measurement is performed at the end. The output state in the real case is thus a sum of only two terms: an accept and reject case. We will again analyze these separately, and will show that the accept state is approximately equal to Equation (4.57), while the reject state approximates Equations (4.59) + (4.60).

Following Protocol 4.4.2 on an input state ρ^{MR}, and canceling out the F_i and $F_i^\dagger$ terms that are part of the honest actions, we first consider the state in case of accept. We abbreviate

$$\begin{aligned} \sigma :=& \mathop{\mathbb{E}}_{g} E^{MT_1} U_g^{T_1T_2}\left(A\left(\rho\otimes|0^{2n}\rangle\langle 0^{2n}|\right)A^\dagger\right)U_g^\dagger E^\dagger & (4.62)\\ =& \quad E^{MT_1}\mathcal{T}_{GL(2n,\mathbb{F}_2)}^{T_1T_2}\left(A\left(\rho\otimes|0^{2n}\rangle\langle 0^{2n}|\right)A^\dagger\right)E^\dagger, & (4.63)\end{aligned}$$

where we are allowed to view $\mathbb{E}_g U_g(\cdot)U_g^\dagger$ as a Twirling operation, since A and B are independent of g. We decompose the attack B as in Equation (4.56), and derive the accept case

$$\begin{aligned}
&\mathop{\mathbb{E}}_{E,r,s} \langle r|^{T_2} B \left(\mathsf{X}^r\mathsf{Z}^s\right)^{T_2} \sigma \left(\mathsf{X}^r\mathsf{Z}^s\right)^\dagger B^\dagger |r\rangle \otimes |E\rangle\langle E|^S && (4.64)\\
&= \mathop{\mathbb{E}}_{E,r,s} \langle 0|^{T_2} \left(\mathsf{X}^r\mathsf{Z}^s\right)^{\dagger T_2} B \left(\mathsf{X}^r\mathsf{Z}^s\right)^{T_2} \sigma \left(\mathsf{X}^r\mathsf{Z}^s\right)^\dagger B^\dagger \left(\mathsf{X}^r\mathsf{Z}^s\right) |0\rangle \otimes |E\rangle\langle E|^S && (4.65)\\
&= \mathop{\mathbb{E}}_{E,r,s} \sum_{b,d,b',d'} \langle 0|^{T_2} \left(\left(\mathsf{X}^r\mathsf{Z}^s\mathsf{X}^b\mathsf{Z}^d\mathsf{X}^r\mathsf{Z}^s\right) \otimes B_{b,d}\right) \sigma \left(\left(\mathsf{X}^r\mathsf{Z}^s\mathsf{X}^{b'}\mathsf{Z}^{d'}\mathsf{X}^r\mathsf{Z}^s\right) \otimes B^\dagger_{b',d'}\right) |0\rangle \\
&\qquad \otimes |E\rangle\langle E|^S && (4.66)\\
&= \mathop{\mathbb{E}}_{E} \sum_{b,d} \langle b|^{T_2} B_{b,d} \sigma B^\dagger_{b,d} |b\rangle \otimes |E\rangle\langle E|^S && (4.67)\\
&= \mathop{\mathbb{E}}_{E} \sum_{b,d} B_{b,d} \mathrm{Tr}_{T_2} \left[\Pi^{T_2}_{b,H} \sigma \Pi^\dagger_{b,H}\right] B^\dagger_{b,d} \otimes |E\rangle\langle E|^S, && (4.68)
\end{aligned}$$

where $\Pi_{b,H}$ is defined in Equation (4.12). From Equation (4.66) to (4.67), we used the Pauli twirl to remove all terms for which $(b,d) \neq (b',d')$. This application of the Pauli twirl is possible, because neither A nor B depends on r, s.

We continue with the accept case by expanding σ in Equation (4.68), and evaluate the effect of the random $\mathrm{GL}_{2n,\mathbb{F}_2}$ element on T_1T_2 using Lemma 4.4.4. It ensures that, if A altered the T_1T_2 register, then B cannot successfully reset the register T_2 to the correct value r. It follows that

$$\begin{aligned}
&\text{Eq. (4.68)} && (4.69)\\
&\approx \mathop{\mathbb{E}}_{E} \sum_{b,d} B^{MT_1R}_{b,d} E^{MT_1} \mathrm{Tr}_{T_2} \left[\Pi^{T_1T_2}_{b,F} A \left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right) A^\dagger \Pi^\dagger_{b,F}\right] E^\dagger B^\dagger_{b,d} \otimes |E\rangle\langle E|^S && (4.70)\\
&= \mathop{\mathbb{E}}_{E} \sum_{d} B^{MT_1R}_{0,d} E^{MT_1} \mathrm{Tr}_{T_2} \left[|0^{2n}\rangle\langle 0^{2n}| A \left(\rho \otimes |0^{2n}\rangle\langle 0^{2n}|\right) A^\dagger |0^{2n}\rangle\langle 0^{2n}|\right] E^\dagger B^\dagger_{0,d} \\
&\qquad \otimes |E\rangle\langle E|^S && (4.71)\\
&= \mathop{\mathbb{E}}_{E} \sum_{d} B^{MT_1R}_{0,d} E^{MT_1} \left(A_0 \rho A_0^\dagger \otimes |0^n\rangle\langle 0^n|\right) E^\dagger B^\dagger_{0,d} && (4.72)\\
&= \text{Eq. (4.57)}. && (4.73)
\end{aligned}$$

The difference in the approximation is bound by $\mathrm{negl}(n)$, since for each b we can use Lemma 4.4.4 (and there is an implicit average over the bs because of the normalization factor induced by the $B_{b,d}$ operator). Essentially, the only way to pass the measurement test successfully is for A not to alter the all-zero state in T_1T_2, and for B to leave T_2 unaltered in the computational basis. This is reflected in the simulator's zero filter and X filter, respectively.

If the real protocol rejects, the MPC stores a dummy $\perp$ in the key register S. The resulting state can be derived in a similar way, up to Equation (4.70), after which the derivation becomes slightly different. The output state in the case of reject approximates (up to a difference of $\text{negl}(n)$)

$$\begin{aligned}
&\mathop{\mathbb{E}}_{E}\sum_{b,d} B_{b,d}^{MT_1R} E^{MT_1} \mathrm{Tr}_{T_2}\left[\left(\mathbb{I}-\Pi_{b,F}\right)^{T_1T_2} A\left(\rho\otimes|0^{2n}\rangle\langle 0^{2n}|\right)A^\dagger\left(\mathbb{I}-\Pi_{b,F}\right)^\dagger\right]E^\dagger B_{b,d}^\dagger \\
&\qquad\qquad \otimes|\perp\rangle\langle\perp|^S && (4.74)\\
=\;& \sum_{b,d} B_{b,d}^{MT_1R}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(\mathrm{Tr}_{T_2}\left[\left(\mathbb{I}-\Pi_{b,F}\right)^{T_1T_2} A\left(\rho\otimes|0^{2n}\rangle\langle 0^{2n}|\right)A^\dagger\left(\mathbb{I}-\Pi_{b,F}\right)^\dagger\right]\right)B_{b,d}^\dagger \\
&\qquad\qquad \otimes|\perp\rangle\langle\perp|^S && (4.75)\\
=\;& \sum_{\substack{b\neq 0^n\\ d,a,a'}} B_{b,d}^{MT_1R}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(\mathrm{Tr}_{T_2}\left[A_a\rho^{MR}A_{a'}^\dagger\otimes|a\rangle\langle a'|\right]\right)B_{b,d}^\dagger\otimes|\perp\rangle\langle\perp|^S \\
&\quad + \sum_{\substack{a,a'\neq 0^{2n}\\ d}} B_{0,d}^{MT_1R}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(\mathrm{Tr}_{T_2}\left[A_a\rho^{MR}A_{a'}^\dagger\otimes|a\rangle\langle a'|\right]\right)B_{0,d}^\dagger\otimes|\perp\rangle\langle\perp|^S && (4.76)\\
=\;& \sum_{\substack{(b,a)\neq(0^n,0^{2n})\\ d}} B_{b,d}^{MT_1R}\left(\mathrm{Tr}_M\left[A_a\rho^{MR}A_a^\dagger\right]\otimes\tau^{MT_1}\right)B_{b,d}^\dagger\otimes|\perp\rangle\langle\perp|^S && (4.77)\\
=\;& \text{Eq. (4.59)}+\text{Eq. (4.60)}. && (4.78)
\end{aligned}$$

Tracing out register T_2 ensures that the second half of a and a' have to be equal; Twirling over the Clifford group ensures that the first half (acting on register T_1) of a and a' have to be equal (see the proof of Lemma 2.4.9).

These derivations show that the output state that the environment sees (in registers MT_1RS) in the real protocol are negligibly close to the output state in the ideal protocol. This concludes our proof for the second case, where player 1 is dishonest. □

4.4.2 Preparing ancilla qubits

Apart from encrypting the players' inputs, we also need a way to obtain encoded ancilla-zero states, which may be fed as additional input to the circuit. Since none of the players can be trusted to simply generate these states as part of their input, we need to treat them separately.

In the two-party protocol [DNS12], Alice generates an encoding of $|0\rangle\langle 0|$, and Bob tests it by entangling (with the help of the classical MPC) the data qubit with a separate $|0\rangle\langle 0|$ qubit. Upon measuring that qubit, Bob then either detects a maliciously generated data qubit, or collapses it into the correct state. For details, see [DNS12, Appendix E].

Here, we take a similar approach, except with a public test on the shared traps. In order to guard against a player that may lie about the measurement outcomes during a test, we entangle the data qubits with *all* traps. We do so using a random linear operator, similarly to the encoding described in the previous subsection.

Essentially, the protocol for preparing ancilla qubits is identical to Protocol 4.4.2 for input encoding, except that now we do not only test whether the $2n$ traps are in the $|0\rangle\langle 0|$ state, but also the data qubit: concretely, the linear operator g acts on $2n+1$ elements instead of $2n$. That is,

$$V := (E \otimes P) U_g (F_k \cdots F_2 F_1)^\dagger. \tag{4.79}$$

As a convention, Player 1 will always create the ancilla $|0\rangle\langle 0|$ states and encode them. In principle, the ancillas can be created by any other player, or by all players together.

Per the same proof as for Lemma 4.4.3, we have implemented the following ideal functionality, again making use of a classical MPC as a black box.

Definition 4.4.7 (Ideal functionality for encoding of $|0\rangle\langle 0|$)**.** Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.

1. The trusted third party initializes a register T_1 with $|0^n\rangle\langle 0^n|$, applies a random $(n+1)$-qubit Clifford E to MT_1, and sends these registers to player 1.

2. All players $i \in I_{\mathcal{A}}$ send a bit b_i to the trusted third party. If $b_i = 0$ for all i, then the trusted third party stores the key E in the state register S of the ideal functionality. Otherwise, it aborts by storing $\perp$ in S.

4.5 Protocol: Clifford computation and measurement

After all players have successfully encoded their inputs and sufficiently many ancillary qubits, they perform a quantum computation gate-by-gate on their joint inputs. In this section, we will present a protocol for circuits that consist only of Clifford gates and computational-basis measurements. The Clifford gates may be classically controlled (for example, on the measurement outcomes that appear earlier in the circuit). In Section 4.6, we will discuss how to expand the protocol to general quantum circuits.

Concretely, we wish to achieve the functionality in Definition 4.2.2 for all circuits C that consist of Clifford gates and computational-basis measurements. As an intermediate step, we aim to achieve the following ideal functionality, where the players only receive an *encoded* output, for all such circuits:

Definition 4.5.1 (Ideal quantum k-party computation without decoding)**.** Let C be a quantum circuit on W wires. Consider a partition of the wires into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. Let $I_{\mathcal{A}} \subsetneq [k]$ be the set of corrupted players.

1. All players i send their register R_i^{in} to the trusted third party.
2. The trusted third party instantiates R^{ancilla} with $|0\rangle\langle 0|$ states.
3. The trusted third party applies C to the wires $[W]$.
4. For every player i and every output wire $w \in R_i^{\mathsf{out}}$, the trusted third party samples a random $(n+1)$-qubit Clifford E_w, applies $\rho \mapsto E_w(\rho \otimes |0^n\rangle\langle 0^n|)E_w^\dagger$ to w, and sends the result to player i.
5. All players $i \in I_{\mathcal{A}}$ send a bit b_i to the trusted third party.
   (a) If $b_i = 0$ for all i, all keys E_w and all measurement outcomes are stored in the state register S.
   (b) Otherwise, the trusted third party `aborts` by storing $\perp$ in S.

To achieve the ideal functionality, we define several subprotocols. The subprotocols for encoding the players' inputs and ancillary qubits have already been described in Section 4.4. It remains to describe the subprotocols for (classically-controlled) single-qubit Clifford gates (Section 4.5.1), (classically controlled) CNOT gates (Section 4.5.2), and computational-basis measurements (Section 4.5.3).

In Section 4.5.5, we show how to combine the subprotocols in order to compute any polynomial-sized Clifford+measurement circuit. Our approach is inductive in the number of gates in the circuit. The base case is the identity circuit, which is essentially covered in Section 4.4. In Sections 4.5.1 to 4.5.3, we show that the ideal functionality for any circuit C, followed by the subprotocol for a gate G, results in the ideal functionality for the circuit $G \circ C$ (C followed by G). As such, we can chain together the subprotocols to realize the ideal functionality in Definition 4.5.1 for any polynomial-sized Clifford+measurement circuit. Combined with the decoding subprotocol we present in Section 4.5.4, such a chain of subprotocols satisfies Definition 4.2.2 for ideal k-party quantum Clifford+measurement computation with abort.

In Definition 4.5.1, all measurement outcomes are stored in the state register of the ideal functionality. We do so to ensure that the measurement results can be used as a classical control to gates that are applied after the circuit C, which can be technically required when building up to the ideal functionality for C inductively. Our protocols can easily be altered to broadcast measurement results as they happen, but the functionality presented in Definition 4.5.1 is the most general: if some player is supposed to learn a measurement outcome m_ℓ, then the circuit can contain a gate X^{m_ℓ} on an ancillary zero qubit that will be part of that player's output.

4.5.1 Subprotocol: single-qubit Cliffords

Due to the structure of the Clifford code, applying single-qubit Cliffords is simple: the classical MPC, who keeps track of the encoding keys, can simply update the key so that it includes the single-qubit Clifford on the data register (see Section 3.4.1). We describe the case of a single-qubit Clifford that is classically controlled on a previous measurement outcome stored in the MPC's state. The unconditional case can be trivially obtained by omitting the conditioning.

Protocol 4.5.2 (Single-qubit Cliffords)**.** Let G^{m_ℓ} be a single-qubit Clifford to be applied on a wire w (held by a player i), conditioned on a measurement outcome m_ℓ. Initially, player i holds an encoding of the state on that wire, and the classical MPC holds the encoding key E.

1. MPC reads result m_ℓ from its state register S, and updates its internally stored key E to $E((G^{m_\ell})^\dagger \otimes \mathsf{I}^{\otimes n})$.

If $m_\ell = 0$, nothing happens. To see that the protocol is correct for $m_\ell = 1$, consider what happens if the state $E(\rho \otimes |0^n\rangle\langle 0^n|)E^\dagger$ is decoded using the updated key: the decoded output is

$$(E(G^\dagger \otimes \mathsf{I}^{\otimes n}))^\dagger E\left(\rho \otimes |0^n\rangle\langle 0^n|\right) E^\dagger (E(G^\dagger \otimes \mathsf{I}^{\otimes n})) \quad = \quad G\rho G^\dagger \otimes |0^n\rangle\langle 0^n|. \tag{4.80}$$

Protocol 4.5.2 implements the ideal functionality securely: given an ideal implementation $\mathfrak{I}^C$ for some circuit C, we can implement $G^{m_\ell} \circ C$ (i.e., the circuit C followed by the gate G^{m_ℓ}) by performing Protocol 4.5.2 right after the interaction with $\mathfrak{I}^C$.

Lemma 4.5.3. *Let G^{m_ℓ} be a single-qubit Clifford to be applied on a wire w (held by a player i), conditioned on a measurement outcome m_ℓ. Let $\Pi^{G^{m_\ell}}$ be Protocol 4.5.2 for the gate G^{m_ℓ}, and $\mathfrak{I}^C$ be the ideal functionality for a circuit C as described in Definition 4.5.1. For all sets $I_\mathcal{A} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_\mathcal{A}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{G^{m_\ell}} \diamond \mathfrak{I}^C)_\mathcal{A})] = \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_\mathcal{S}^{G^{m_\ell} \circ C})].$$

Proof. For the sake of clarity, assume again that there is only one wire, held by player 1 (who might be honest or dishonest). Generalizing the proof to multiple wires does not require any new technical ingredients, but simply requires a lot more (cluttering) notation.

In the protocol $\Pi^{G^{m_\ell}} \diamond \mathfrak{I}^C$, an adversary has two opportunities to attack: once before its input state is submitted to $\mathfrak{I}^C$, and once afterwards. We define a simulator that applies these same attacks, except that it interacts with the ideal functionality $\mathfrak{I}^{G^{m_\ell} \circ C}$.

The adversary $\mathcal{A}$ receives a state ρ^{MR} from the environment (where again, $M := R_1^{\mathsf{in}}$). It potentially alters this state with a unitary map A, submits the result to the ideal functionality, and receives the register $MT_1 = R_1^{\mathsf{out}}$. The adversary may again act on the state, say with a map B, and then gets a chance to submit (for all players $i \in I_{\mathcal{A}}$) bits b_i to $\mathfrak{I}^C$, and b'_i to $\Pi^{G^{m_\ell}}$). If one or more of those bits are 1, the ideal functionality (or the MPC) aborts by overwriting the state register S with $\perp$.

In case all bits are 0, the output register MT_1RS contains

$$\begin{aligned}
&\mathop{\mathbb{E}}_{E} B^{MT_1R} E^{MT_1}\left(\Phi_C^M\left(A\rho^{MR}A^\dagger\right)\otimes|0^n\rangle\langle 0^n|^{T_1}\right)E^\dagger B^\dagger \\
&\qquad\otimes\left|E\left(\left(G^{m_\ell}\right)^\dagger\otimes\mathsf{I}^{\otimes n}\right)\right\rangle\left\langle E\left(\left(G^{m_\ell}\right)^\dagger\otimes\mathsf{I}^{\otimes n}\right)\right|^S \qquad (4.81)\\
=&\mathop{\mathbb{E}}_{E} B^{MT_1R} E^{MT_1}\left(G^{m_\ell}\right)^M\left(\Phi_C^M\left(A\rho^{MR}A^\dagger\right)\otimes|0^n\rangle\langle 0^n|^{T_1}\right)\left(G^{m_\ell}\right)^\dagger E^\dagger B^\dagger\otimes|E\rangle\langle E|^S, \qquad (4.82)
\end{aligned}$$

where $\Phi_C(\cdot)$ is the map induced by the circuit C.

In case not all bits are 0, the output register MT_1RS contains

$$B'A'\rho_R(B'A')^\dagger\otimes\tau^{MT_1}\otimes|\perp\rangle\langle\perp|^S, \qquad (4.83)$$

where A' and B' are the reduced maps A and B on register R.

Define a simulator $\mathcal{S}$ as follows:

Simulator 4.5.4. On input ρ^{MR} from the environment, do:

- Run A on MR.
- Submit M to the ideal functionality for $G^{m_\ell}\circ C$, and receive MT_1.
- Run B on MT_1R, and note its output bits (b_i, b'_i) for all $i \in I_{\mathcal{A}}$. Submit $\max\{b_i, b'_i\}$ to the ideal functionality for $G^{m_\ell}\circ C$.

From the point of view of the adversary, the state it receives from the ideal functionality is the same: a Clifford-encoded state. Thus, the bits b_i and b'_i will not be different in this simulated scenario. In fact, the output state is exactly Eq. (4.82) + Eq. (4.83). □

4.5.2 Subprotocol: CNOT gates

The application of two-qubit Clifford gates (such as CNOT) is more complicated than the single-qubit case, for two reasons.

First, a CNOT is a *joint* operation on two states that are encrypted with *separate* keys. If we were to classically update two keys E_1 and E_2 in a similar fashion as in Protocol 4.5.2, we would end up with a new key $(E_1\otimes E_2)(\mathsf{CNOT}_{1,n+2})$, which cannot be written as a product of two separate keys. The keys would become “entangled”, which is undesirable for the rest of the computation.

Second, the input qubits might belong to separate players, who may not trust the authenticity of each other’s qubits. Dupuis et al. guarantee authenticity of the output state by having both players test each state several times [DNS12]. In a multi-party setting, both players involved in the CNOT are potentially dishonest, so it might seem necessary to involve all players in this extensive testing. However, because all our tests are publicly verified, our protocol requires less testing. Still, interaction with all other players is necessary to apply a fresh “joint” Clifford on the two ciphertexts.

Protocol 4.5.5 (CNOT)**.** This protocol applies a CNOT gate to wires w_i (control) and w_j (target), conditioned on a measurement outcome m_ℓ. Suppose that player i holds an encoding of the first wire, in register $M^i T_1^i$, and player j of the second wire, in register $M^j T_1^j$. The classical MPC holds the encoding keys E_i and E_j.

1. If $i \neq j$, player j sends their registers $M^j T_1^j$ to player i. Player i now holds a $(2n+2)$-qubit state.
2. Player i initializes the registers T_2^i and T_2^j both in the state $|0^n\rangle\langle 0^n|$.
3. For all players h, MPC samples random $(4n+2)$-qubit Cliffords D_h, and gives them to the respective players. Starting with player i, each player h applies D_h to $M^{ij} T_{12}^{ij}$,[a] and sends the state to player $h+1$. Eventually, player i receives the state back from player $i-1$. MPC remembers the applied Clifford
$$D := D_{i-1} D_{i-2} \cdots D_1 D_k D_{k-1} \cdots D_i\,.$$
4. MPC samples random $(2n+1)$-qubit Cliffords F_i and F_j, and tells player i to apply
$$V := (F_i \otimes F_j)\mathsf{CNOT}_{1,2n+2}^{m_\ell}(E_i^\dagger \otimes \mathsf{I}^{\otimes n} \otimes E_j^\dagger \otimes \mathsf{I}^{\otimes n})D^\dagger.$$
Here, the CNOT acts on the two data qubits inside the encodings.
5. If $i \neq j$, player i sends $M^j T_{12}^j$ to player j.
6. Players i and j publicly test their encodings. The procedures are identical, we describe the steps for player i:

(a) MPC samples a random $(n+1)$-qubit Clifford E_i', which will be the new encoding key. Furthermore, MPC samples random n-bit strings s_i and r_i, and a random classical invertible linear operator g_i on $\mathbb{F}_2^{2n}$.

(b) MPC tells player i to apply

$$W_i := \left(E_i' \otimes (\mathsf{X}^{r_i}\mathsf{Z}^{s_i})^{T_2^i}\right) U_{g_i}^{T_{12}^i} F_i^\dagger.$$

Here, U_{g_i} is as defined in Protocol 4.4.2.

(c) Player i measures T_2^i in the computational basis and reports the n-bit measurement outcome r_i' to the MPC.

(d) MPC checks whether $r_i' = r_i$. If it is not, MPC sends `abort` to all players. If it is, the test has passed, and MPC stores the new encoding key E_i' in its internal memory.

[a]We combine subscripts and superscripts to denote multiple registers: e.g., T_{12}^{ij} is shorthand for $T_1^i T_2^i T_1^j T_2^j$.

Lemma 4.5.6. *Let $\Pi^{\mathsf{CNOT}^{m_\ell}}$ be Protocol 4.5.5, to be executed on wires w_i and w_j, held by players i and j, respectively. Let $\mathfrak{I}^C$ be the ideal functionality for a circuit C as described in Definition 4.5.1. For all sets $I_\mathcal{A} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_\mathcal{A}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\mathsf{CNOT}^{m_\ell}} \diamond \mathfrak{I}^C)_\mathcal{A})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_\mathcal{S}^{\mathsf{CNOT}^{m_\ell} \circ C})]\right| \leq \mathrm{negl}(n).$$

Proof. There are four different cases, depending on which of the players i and j are dishonest: both players involved in the CNOT are honest ($i, j \notin I_\mathcal{A}$), both players are dishonest ($i, j \in I_\mathcal{A}$), only player i is honest ($i \notin I_\mathcal{A}$, $j \in I_\mathcal{A}$), or only player j is honest ($i \in I_\mathcal{A}$, $j \notin I_\mathcal{A}$). Without loss of generality, we will assume that all other players are dishonest (except in the second case, where at least one of the other players has to be honest), and that they have no inputs themselves: their encoded inputs can be regarded as part of the adversary's side information R. Note that these four cases also cover the possibility that $i = j$. We describe simulators separately for all four cases.

Case 1: player i and j are honest. In this case, the adversarial players in $I_\mathcal{A}$ only have influence on the execution of Step 3 of the protocol, where the state is sent around in order for the players to jointly apply the random Clifford D.

As in the first case of the proof of Lemma 4.4.3 for the encoding protocol Π^{Enc} (where the encoding player is honest), define a simulator that performs a Pauli filter

IdFilter on the attack of the adversary. The simulator and proof are almost identical to those in Lemma 4.4.3, so we omit the details.

Case 2: player i and j are dishonest. The proof for this case is the most involved, so we start off with an intuitive description of why the protocol is secure against dishonest players i and j.

There are a few ways in which the adversary may attack. First, he may prepare a nonzero state in the registers T_2^i (or T_2^j) in Step 2, potentially intending to spread those errors into $M^i T_1^i$ (or $M^j T_1^j$). Doing so, however, will cause U_{g_i} (or U_{g_j}) to map the trap state to a random nonzero string, and the adversary would not know what measurement string r'_i (or r'_j) to report. Since g_i is unknown to the adversary, Lemma 4.4.4 is applicable in this case: it states that it suffices to measure T_2^i in order to detect any errors in T_{12}^i.

Second, the adversary may fail to execute its instructions V or $W_i \otimes W_j$ correctly. Doing so is equivalent to attacking the state right before or right after these instructions. In both cases, however, the state in $M^i T_1^i$ is Clifford-encoded (and the state in T_2^i is Pauli-encoded) with keys unknown to the adversary, so the authentication property of the Clifford code prevents the adversary from altering the outcome.

The simulator we will define tests the adversary exactly for the types of attacks above. With Pauli filters, the simulator checks whether the attacker leaves the authenticated states and the trap states T_2^i and T_2^j (both at initialization and before measurement) unaltered.

We now analyze the security in more detail. Without loss of generality, we can break up the attack of the adversary (acting jointly for players i, j and any other players in $I_{\mathcal{A}}$) into three unitary operations, acting on the relevant register plus a side-information register. As in the proof of Lemma 4.4.3, we may assume that the honest actions are executed as well, since each attack may start or end with undoing that honest action. The first attack $A^{M^{ij}R}$ is executed on the plaintexts, before any protocol starts. The second attack $\tilde{A}^{M^{ij}T_{12}^{ij}R}$ happens after Step 2 of Protocol 4.5.5, on the output of $\mathfrak{I}^C$ and the initialized registers T_2^{ij}. Finally, the third attack $\tilde{\tilde{A}}^{M^{ij}T_{12}^{ij}R}$ happens toward the end of the protocol, right before the T_2^{ij} registers are measured in Step 6c of Protocol 4.5.5. Note that $\tilde{\tilde{A}}$ may depend on the instructions V, W_i and W_j.

It will be useful to decompose the second and third attacks as follows:

$$\begin{aligned}\tilde{A} = \sum_{a_1^i,a_2^i,a_1^j,a_2^j,c_1^i,c_2^i,c_1^j,c_2^j} & (\mathsf{X}^{a_1^i}\mathsf{Z}^{c_1^i})^{M^iT_1^i} \otimes (\mathsf{X}^{a_2^i}\mathsf{Z}^{c_2^i})^{T_2^i} \\ & \otimes (\mathsf{X}^{a_1^j}\mathsf{Z}^{c_1^j})^{M^jT_1^j} \otimes (\mathsf{X}^{a_2^j}\mathsf{Z}^{c_2^j})^{T_2^j} \otimes \tilde{A}^R_{a_{12}^{ij},c_{12}^{ij}} \end{aligned} \tag{4.84}$$

$$\tilde{\tilde{A}} = \sum_{b,d} (\mathsf{X}^b\mathsf{Z}^d)^{T_2^{ij}} \otimes \tilde{\tilde{A}}_{b,d}^{M^{ij}T_1^{ij}R} \tag{4.85}$$

Whenever the order is clear from the context, we will abbreviate, for example, a_{12}^{ij} for the concatenation $a_1^i a_2^i a_1^j a_2^j$, as we have done in the last term of Equation (4.84).

In terms of an arbitrary attack $A, \tilde{A}, \tilde{\tilde{A}}$, define the simulator $\mathcal{S}$ as follows:

Simulator 4.5.7. On input $\rho^{M^i M^j R}$ from the environment, do:

1. Initialize $b_i = 0$.
2. Run A on $M^{ij}R$.
3. Submit M^{ij} to $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell} \circ C}$, and receive $M^{ij}T_1^{ij}$, containing an encodings of the M^i and M^j registers of $\mathsf{CNOT}^{m_\ell}(C(\rho))$, under some (secret) keys E_i, E_j.
4. Run $\mathsf{ZeroFilter}^{T_2^{ij}}(\mathsf{IdFilter}^{M^{ij}T_1^{ij}}(\tilde{A}))$ on R (see Section 4.3.1). If one of the filter flags is 1, set $b_i = 1$.
5. Sample random $V' \in \mathscr{C}_{4n+2}$ and $W_i', W_j' \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^{ij}}(\tilde{\tilde{A}})$ on $M^{ij}T_1^{ij}R$, where $\tilde{\tilde{A}}$ may depend on V', W_i', W_j'. If the filter flag is 1, set $b_i = 1$.
6. Submit b_i to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_i\}$.

The simulator should also abort whenever the adversary signals abort during an interaction with MPC. For simplicity, we leave out these abort bits in the simulator and proof. They are dealt with in the same way as in the proof of Lemma 4.5.3.

As before, we derive the real and ideal output states in the registers $R_i^{\mathsf{out}} = M^i T_1^i$ and $R_j^{\mathsf{out}} = M^j T_i^j$, the state register S, and the attacker's side information R, and aim to show that they are negligibly close in terms of the security parameter n.

In the ideal (simulated) case, there are two points at which cheating may be detected by the simulator: once during the zero/identity filter of $\tilde{A}$, and during the X filter of $\tilde{\tilde{A}}$. Thus, there are three possible outcome scenarios: both tests are passed, the first test is passed but the second is not, or the first test fails (in which case it does not matter whether the second test is passed or not).

If both tests pass, then by three applications of Lemma 4.3.2, the simulated output state is

$$\mathop{\mathbb{E}}_{E_i,E_j} \tilde{\tilde{A}}_0^{M^{ij}T_1^{ij}R} \tilde{A}_{0,0}^{R} (E_i \otimes E_j)^{M^{ij}T_1^{ij}} \left(\mathsf{CNOT}^{m_\ell} \mathcal{C}\left(A\rho A^\dagger\right) \mathsf{CNOT}^{m_\ell \dagger} \otimes |0^{2n}\rangle\langle 0^{2n}|^{T_1^{ij}}\right)$$
$$(E_i \otimes E_j)^\dagger \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_0^\dagger \otimes |E_i, E_j\rangle\langle E_i, E_j|^S, \tag{4.86}$$

where we write $\tilde{A}_{0,0}$ to denote the attack $\sum_{c_2^{ij}} \tilde{A}_{0000,0c_2^i0c_2^j}$ that passes through the zero/identity filter, and $\tilde{\tilde{A}}_0$ to denote the attack $\sum_d \tilde{\tilde{A}}_{0,d}$ that passes through the X filter.

If the first test is passed but the second test is not, then the storage register S gets erased, so that we may view the E_i and E_j operations as Clifford twirls of the registers they encode. In that case, the (simulated) output state is

$$\sum_{b\neq 0} \tilde{\tilde{A}}_b \tilde{A}_{0,0} \mathcal{T}_{\mathscr{C}_{n+1}}^{M^i T_1^i} \left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^j T_1^j} \left(\mathsf{CNOT}^{m_\ell} \mathcal{C}\left(A\rho A^\dagger\right) \mathsf{CNOT}^{m_\ell \dagger} \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right)\right) \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_b^\dagger \\ \otimes |\bot\rangle\langle\bot|^S \tag{4.87}$$

$$= \sum_{b\neq 0} \tilde{\tilde{A}}_b \tilde{A}_{0,0} \left(\mathrm{Tr}_{M^{ij}} \left[A\rho^{M^{ij}R} A^\dagger \right] \otimes \tau^{MT_1^{ij}} \right) \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_b^\dagger \otimes |\bot\rangle\langle\bot|^S . \tag{4.88}$$

The Clifford twirls cause the data and trap registers to become fully mixed, thereby also nullifying the effect of the CNOT and circuit C on the data.

Finally, we consider the third scenario, where already the first test (the zero / identity filter) fails. As in the previous scenario, the storage register S is erased, allowing us to apply the Clifford twirl again. By Lemma 4.3.2, the output state in this case is

$$\sum_b \sum_{\substack{(a_{12}^{ij}, c_1^{ij}) \neq \\ (0^{4n+2}, 0^{2n+2})}} \tilde{\tilde{A}}_b \tilde{A}_{a_{12}^{ij}, c_1^{ij}} \left(\mathrm{Tr}_{M^{ij}} \left[A\rho^{M^{ij}R} A^\dagger \right] \otimes \tau^{MT_1^{ij}} \right) \tilde{A}^\dagger_{a_{12}^{ij}, c_1^{ij}} \tilde{\tilde{A}}_b^\dagger \otimes |\bot\rangle\langle\bot|^S , \tag{4.89}$$

writing $\tilde{A}_{a_{12}^{ij}, c_1^{ij}} := \sum_{c_2^{ij}} \tilde{A}_{a_{12}^{ij}, c_{12}^{ij}}$. Note that for the second test (the X filter), the terms for both possible flag values remain: the cheating bit b_i is already set to 1, regardless of the outcome of this second test.

We move on to the analysis of the real protocol $\Pi^{\mathsf{CNOT}^{m_\ell}} \diamond \mathfrak{I}^C$, and aim to show that the output state is equal to Eq. (4.86) + Eq. (4.88) + Eq. (4.89). To do so, consider the output state of the real protocol, right before the final measurement.

We continue to argue why the attacks are independent of E_{ij}, E'_{ij}, g_{ij}, r_{ij} and s_{ij}. The intuition for this fact is that D is uniformly random and independent of E_{ij} from the perspective of the adversary. Therefore it "hides" all other information that is used to compile V, including F_{ij}. Therefore F_{ij} are as random and independent as D from the perspective of the adversary, i.e., given V. This allows for a similar argument for the Cliffords W_{ij}, where F hides all the other information, i.e., E'_{ij}, g_{ij}, r_{ij} and s_{ij}.

For the following more formal argument, we treat all the mentioned quantities as random variables. Initially, E_{ij} are uniformly random. D is the product of a number of Clifford group elements, at least one of which is generated honestly and therefore sampled uniformly at random. But for any group G, given two independent random variables ζ and η on G, where ζ is uniformly random, we have that $\eta\zeta$ is uniformly random and $\eta\zeta\perp\eta$, where $\perp$ denotes independence. This implies that D is indeed a uniformly random Clifford itself. Using the same argument, V is uniformly random and $V\perp(E_{ij},F_{ij})$. The quantities E'_{ij}, g_{ij}, r_{ij} and s_{ij} are sampled independently and uniformly after V is handed to player i, so we even have $V\perp(E_{ij},F_{ij},E'_{ij},g_{ij},r_{ij},s_{ij})$. After Step 4 in Protocol 4.5.5, the adversary has a description of V, so when analyzing W_{ij}, we have to derive independence statements *given* V. But as shown before F_{ij} are independent of V, so the the group random variable property above we have $W_{ij}\perp(E'_{ij},g_{ij},r_{ij},s_{ij})|V$. Clearly, E_{ij} is independent of all the random variables used in W_{ij}, and we have shown that $E_{ij}\perp V$, so $W_{ij}\perp(E_{ij},E'_{ij},g_{ij},r_{ij},s_{ij})|V$. In summary,

$$(V,W_{ij})\perp(E_{ij},E'_{ij},g_{ij},r_{ij},s_{ij}). \tag{4.90}$$

According to the decomposition of the attack into attack maps A, $\tilde{A}$ and $\tilde{\tilde{A}}$, that we made without loss of generality, the Clifford operations F_i, F_j, and D cancel again after having fulfilled their task of hiding information, which allows us to utilize Equation (4.90) to carry out the expectation values over various variables from the right hand side of that equation.

The output state of the real protocol is

$$\mathop{\mathbb{E}}_{\substack{E'_i,E'_j,g_i,g_j\\ r_i,s_i,r_j,s_j}} \tilde{\tilde{A}}^{M^{ij}T_{12}^{ij}R}\left(E'_i\otimes\left(\mathsf{X}^{r_i}\mathsf{Z}^{s_i}\right)^{T_2^i}\otimes E'_j\otimes\left(\mathsf{X}^{r_j}\mathsf{Z}^{s_j}\right)^{T_2^j}\right)\left(U_{g_i}^{T_{12}^i}\otimes U_{g_j}^{T_{12}^j}\right)\mathsf{CNOT}^{m_\ell}\sigma$$
$$\mathsf{CNOT}^{m_\ell\dagger}\left(U_{g_i}^\dagger\otimes U_{g_j}^\dagger\right)\left(E'_i\otimes\left(\mathsf{X}^{r_i}\mathsf{Z}^{s_i}\right)\otimes E'_j\otimes\left(\mathsf{X}^{r_j}\mathsf{Z}^{s_j}\right)\right)^\dagger\tilde{\tilde{A}}^\dagger\otimes\left|E'_i,E'_j\right\rangle\!\left\langle E'_i,E'_j\right|^S, \tag{4.91}$$

where (again writing $\Phi_C(\cdot)$ for the map induced by the circuit C)

$$\sigma:=\mathop{\mathbb{E}}_{E_i,E_j\in\mathscr{C}_{n+1}}\left(E_i^\dagger\otimes E_j^\dagger\right)\tilde{A}^{M^{ij}T_{12}^{ij}R}\left(\left(E_i^{M^iT_1^i}\otimes E_j^{M^jT_1^j}\right)\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_1^{ij}}\right)\right.$$
$$\left.\left(E_i^\dagger\otimes E_j^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_2^{ij}}\right)\tilde{A}^\dagger\left(E_i\otimes E_j\right) \tag{4.92}$$
$$=\mathcal{T}_{\mathscr{C}_{n+1}}^{M^iT_1^i}\left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^jT_1^j}\left(\tilde{A}\right)\right)\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{4n}\rangle\!\langle0^{4n}|^{T_{12}^{ij}}\right) \tag{4.93}$$
$$\approx_{\mathrm{negl}(n)}\tilde{A}_{00}^{T_2^{ij}R}\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_2^{ij}}\right)\tilde{A}_{00}^\dagger\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_1^{ij}}$$
$$+\mathrm{Tr}_{M^i}\left[\tilde{A}_{01}^{T_2^{ij}R}\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_2^{ij}}\right)\tilde{A}_{01}^\dagger\right]\otimes\tau^{M^iT_1^i}\otimes|0^n\rangle\!\langle0^n|^{T_1^j}$$
$$+\mathrm{Tr}_{M^j}\left[\tilde{A}_{10}^{T_2^{ij}R}\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_2^{ij}}\right)\tilde{A}_{10}^\dagger\right]\otimes|0^n\rangle\!\langle0^n|^{T_1^i}\otimes\tau^{M^jT_1^j}$$
$$+\mathrm{Tr}_{M^{ij}}\left[\tilde{A}_{11}^{T_2^{ij}R}\left(\Phi_C\left(A\rho^{M^{ij}R}A^\dagger\right)\otimes|0^{2n}\rangle\!\langle0^{2n}|^{T_2^{ij}}\right)\tilde{A}_{11}^\dagger\right]\otimes\tau^{M^{ij}T_1^{ij}}, \tag{4.94}$$

and

$$\tilde{A}_{pq} := \sum_{\substack{a_2^{ij}, c_2^{ij} \\ a_1^i c_1^i \in S_p \\ a_1^j c_1^j \in S_q}} (\mathsf{X}^{a_2^{ij}} \mathsf{Z}^{c_2^{ij}})^{T_2^{ij}} \otimes \tilde{A}^R_{a_{12}^i a_{12}^j, c_{12}^i c_{12}^j}. \tag{4.95}$$

for $p, q \in \{0,1\}$ and $S_0 := \{0^{2n+2}\}$, $S_1 := \{0,1\}^{2n+2} \setminus S_0$. The approximation follows by a double application of Lemma 3.3.2. We can twirl with the keys E_i and E_j, since none of the attacks can depend on the secret encoding keys E_i, E_j, and the keys have been removed from the storage register S, and replaced by the new keys E'_i, E'_j.

Having rewritten the state σ in this form, we consider the state in Equation (4.91) after the T_2^{ij} registers are measured in the computational basis, as in Step 6c of Protocol 4.5.5. We first consider the case where the measurement outcome is accepted by the MPC (i.e., the measurement outcome is $r_i r_j$). Using the same derivation steps as in Equations (4.64) to (4.75), we see that the real accept state approximates (up to a negligible error in n)

$$\mathop{\mathbb{E}}_{E'_i, E'_j} \tilde{\tilde{A}}_0^{M^{ij} T_1^{ij} R} \left(E'_i \otimes E'_j\right)^{M^{ij} T_1^{ij}} \mathrm{Tr}_{T_2^{ij}} \left[\left|0^{4n}\right\rangle\!\left\langle 0^{4n}\right|^{T_{12}^{ij}} \mathsf{CNOT}^{m_\ell} \sigma \mathsf{CNOT}^{m_\ell \dagger} \left|0^{4n}\right\rangle\!\left\langle 0^{4n}\right| \right]$$
$$\left(E'_i \otimes E'_j\right)^\dagger \tilde{\tilde{A}}_0^\dagger \otimes \left|E'_i, E'_j\right\rangle\!\left\langle E'_i, E'_j\right|^S. \tag{4.96}$$

To derive the above expression, we applied a Pauli twirl, which relies on the fact that the adversary cannot learn r_i, r_j, s_i, s_j. Furthermore, the derivation contains an application of Lemma 4.4.4 to expand the effect of measuring T_2^{ij} to measuring both registers T_{12}^{ij}. To apply this lemma, we use the aforementioned fact that g_i and g_j remain hidden from the adversary.

The second, third, and fourth terms of the sum in the approximation of σ (see Equation (4.94)) have negligible weight inside Equation (4.96), since the probability of measuring an all-zero string in the T_1^{ij} registers is negligible in n whenever one or both are in the fully mixed state τ. Additionally, the only components in $\tilde{A}_{00}$ that survive are those that act trivially in the computational basis on T_2^{ij}. Hence,

$$\text{Eq. (4.96)} \approx_{\mathrm{negl}(n)} \text{Eq. (4.86)}. \tag{4.97}$$

In case the measurement outcome is rejected by the MPC (i.e., it is anything other than $r_i r_j$), the output state can be derived using the same steps that were used to obtain Equation (4.75) in the proof of Lemma 4.4.3. Up to an error negligible in n, it approximates

$$\sum_b \tilde{\tilde{A}}_b^{M^{ij} T_1^{ij} R} \mathcal{T}_{\mathscr{C}_{n+1}}^{M^i T_1^i} \left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^j T_1^j} \left(\mathrm{Tr}_{T_2^{ij}} \left[(\mathbb{I} - \Pi_{b,F})^{T_{12}^{ij}} \mathsf{CNOT}^{m_\ell} \sigma \mathsf{CNOT}^{m_\ell \dagger} (\mathbb{I} - \Pi_{b,F})^\dagger \right] \right) \right) \tilde{\tilde{A}}_b^\dagger$$
$$\otimes \left|\perp\right\rangle\!\left\langle\perp\right|^S. \tag{4.98}$$

The encoding under the keys E'_i, E'_j in Equation (4.91) can be regarded as two Clifford twirls, because these keys are removed from the storage register S, and because the attack maps also cannot depend on them, since they are unknown by the adversary.

The next step is to substitute the expression for σ that was derived in Equation (4.94). We distinguish between the case $b \neq 0$, where $\mathbb{I} - \Pi_{b,F} = \mathbb{I}$ and thus all terms of Equation (4.94) remain, and the case $b = 0$, where one has to more carefully count which (parts of the) terms remain. To do so, observe that the first term is projected to nonzero in T_{12}^{ij} whenever ${a_2}^{ij}$ is nonzero. The other three terms are always projected to nonzero, up to a negligible contribution of the all-zero string in the fully mixed state τ. In summary, exactly those terms $\tilde{A}^{R}_{a_{12}^{ij}, c_{12}^{ij}}$ remain for which $(a_{12}^{ij}, c_1^{ij}) \neq (0^{4n+2}, 0^{2n+1})$.

Because the two Clifford twirls map the M^{ij} registers to a fully mixed state, the four terms in Equation (4.94) can be combined, resulting in the following output state in the reject case

$$\sum_{b\neq 0} \sum_{a_{12}^{ij}, c_1^{ij}} \tilde{\tilde{A}}_b \tilde{A}_{a_{12}^{ij}, c_1^{ij}} \left(\mathrm{Tr}_{M^{ij}} \left[A\rho^{M^{ij}R} A^\dagger \right] \otimes \tau^{MT_1^{ij}} \right) \tilde{A}^\dagger_{a_{12}^{ij}, c_1^{ij}} \tilde{\tilde{A}}^\dagger_b \otimes |\bot\rangle\langle\bot|^S$$

$$+ \sum_{\substack{(a_{12}^{ij}, c_1^{ij}) \neq \\ (0^{4n+2}, 0^{2n+2})}} \tilde{\tilde{A}}_0 \tilde{A}_{a_{12}^{ij}, c_1^{ij}} \left(\mathrm{Tr}_{M^{ij}} \left[A\rho^{M^{ij}R} A^\dagger \right] \otimes \tau^{MT_1^{ij}} \right) \tilde{A}^\dagger_{a_{12}^{ij}, c_1^{ij}} \tilde{\tilde{A}}^\dagger_0 \otimes |\bot\rangle\langle\bot|^S \tag{4.99}$$

$$= \text{Eq. (4.88)} + \text{Eq. (4.89)}. \tag{4.100}$$

We have shown that the sum of the three terms of the output state in the simulated case (both tests accept, the first test accepts but the second rejects, and the first test rejects) is approximately equal to the sum of the two terms of the output state in the real case (the MPC accepts the measurement outcome, or the MPC rejects the measurement outcome).

Case 3: only player i is honest. At first, it may seem that this is just a special case of the previous one, where both players are dishonest. While this is true in spirit, we cannot directly use the simulator from the previous case. The reason is syntactical: a simulator would not have access to the registers $M^i T_{12}^i$, because they are held by honest player i. Thus, the simulator needs to differ slightly from the previous case. However, it is very similar, as is the derivation of the real/ideal output states. We therefore omit the full proof, and instead only define the simulator.

The adversary again has three opportunities to attack: an attack A on the plaintext and side-information register $M^j R$, which happens before the ideal functionality $\mathfrak{I}^C$ is called; an attack $\tilde{A}$ on the output of $\mathfrak{I}^C$ in registers $M^j T_1^j R$ (right before player j sends their state to player i); and an attack $\tilde{\tilde{A}}$ on $M^j T_{12}^j R$, after an honest application of W_j (which we may assume to happen without loss of generality), but before the computational-basis measurement of T_2. Given these attacks, define the simulator as follows.

Simulator 4.5.8. On input $\rho^{M^j R}$ from the environment, do:

1. Initialize $b_j = 0$.
2. Run A on $M^j R$.
3. Submit M^j to the ideal functionality $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell} \circ C}$, and receive $M^j T_1^j$, containing an encoding under a secret key E_j. (Honest player i holds the other output, encoded under E_i.)
4. Run $\mathsf{IdFilter}^{M^j T_1^j}(\tilde{A})$ on R. If the filter flag is 1, then set $b_j = 1$.
5. Sample random $W_j' \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^j}(\tilde{\tilde{A}})$ on $M^j T_1^j R$, where $\tilde{\tilde{A}}$ may depend on W_j'. If the filter flag is 1, then set $b_j = 1$.
6. Submit b_j to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_j\}$.

Intuitively, the simulator tests whether player j sent the actual outcome of $\mathfrak{I}^C$ without altering it (Step 4 of the simulator), and whether player j left the computational basis of T_2 invariant before measuring it (Step 5 of the simulator).

Case 4: only player j is honest. Similarly to the previous case, we need to provide a separate simulator for the case where player i is dishonest, player j is honest, and (without loss of generality) all other players are dishonest.

The adversary has three opportunities to attack: an attack A on the plaintext and side-information register $M^i R$, which happens before the ideal functionality $\mathfrak{I}^C$ is called; an attack $\tilde{A}$ on registers $M^{ij} T_{12}^{ij} R$ that is applied on the outputs of the ideal functionality and on the extra registers T_2, right before D is applied; and an attack $\tilde{\tilde{A}}$ on $M^{ij} T_{12}^{ij} R$, right before the measurement on T_2^i (as part of player i's test) and the application of W_j (so right before sending the appropriate registers to player j). Given these attacks, define the simulator as follows.

Simulator 4.5.9. On input $\rho^{M^i R}$ from the environment, do:

1. Initialize $b_i = 0$.
2. Run A on $M^i R$.

3. Submit M^i to the ideal functionality $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell} \circ C}$, and receive $M^i T_1^i$, containing an encoding under a secret key E_i. (Honest player j holds the other output, encoded under E_j.)
4. Run $\mathsf{ZeroFilter}^{T_2^{ij}}\left(\mathsf{IdFilter}^{M^{ij} T_1^{ij}}(\tilde{A})\right)$ on R. If the filter flag is 1, then set $b_i = 1$.
5. Sample random $V, W_i' \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^i}\left(\mathsf{IdFilter}^{M^j T_{12}^j}\left(\tilde{\tilde{A}}\right)\right)$ on $M^i T_1^j R$, where $\tilde{\tilde{A}}$ may depend on V and W_i'. If the filter flag is 1, then set $b_i = 1$.
6. Submit b_i to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_i\}$.

Intuitively, the simulator tests (in Step 4) whether player i leaves the states received from the ideal functionality and player j intact, as well as the traps in T_2^{ij} that are initialized to $|0^{2n}\rangle\langle 0^{2n}|$. In Step 5, it tests both whether player i executes the test honestly by not altering the computational-basis value of T_2^i, and whether he would give the correct (uncorrupted) state to player j.

□

4.5.3 Subprotocol: measurement

Measurement of authenticated states introduces a new conceptual challenge. For a random key E, the result of measuring $E\left(\rho \otimes |0^n\rangle\langle 0^n|\right) E^\dagger$ in a fixed basis is in no way correlated with the logical measurement outcome of the state ρ. However, the measuring player is also not allowed to learn the key E, so they cannot perform a measurement in a basis that depends meaningfully on E.

Dupuis et al. solve this challenge by entangling the state with an (encoded) ancilla-zero state on a logical level [DNS10, Appendix E]. After this entanglement step, Alice gets the original state while Bob gets the ancilla state. They both decode their state (learning the key from the MPC), and can measure it. Because those states are entangled, and at least one of Alice and Bob is honest, they can ensure that the measurement outcome was not altered, simply by checking that they both obtained the same outcome. The same strategy can in principle also be scaled up to k players, by making all k players hold part of a big (logically) entangled state. However, doing so requires the application of $k-1$ logical CNOT operations, making it a relatively expensive procedure.

We take a different approach in our protocol. The player that performs the measurement essentially entangles, with the help of the MPC, the data qubit with a random subset of the traps. The MPC later checks the consistency of the outcomes: all entangled qubits should yield the same measurement result.

Our alternative approach has the additional benefit that the measurement outcome can be kept secret from some or all of the players. In the description of the protocol below, the MPC stores the measurement outcome in its internal state. This allows the MPC to classically control future gates on the outcome. If it is desired to instead reveal the outcome to one or more of the players, this can easily be done by performing a classically-controlled X operation on some unused output qubit of those players.

Protocol 4.5.10 (Computational-basis measurement)**.** Player i holds an encoding of the state in a wire w in the register MT_1. The classical MPC holds the encoding key E in the register S.

1. MPC samples random strings $r, s \in \{0,1\}^{n+1}$ and $c \in \{0,1\}^n$.
2. MPC tells player i to apply
$$V := \mathsf{X}^r \mathsf{Z}^s \mathsf{CNOT}_{1,c} E^\dagger$$
to the register MT_1, where $\mathsf{CNOT}_{1,c}$ denotes the unitary $\prod_{i\in[n]} \mathsf{CNOT}_{1,i}^{c_i}$ (that is, the string c dictates with which of the qubits in T_1 the M register will be entangled).
3. Player i measures the register MT_1 in the computational basis, reporting the result r' to MPC.
4. MPC checks whether $r' = r \oplus (m, m \cdot c)$ for some $m \in \{0,1\}$.[a] If so, it stores the measurement outcome m in the state register S. Otherwise, it aborts by storing $\perp$ in S.
5. MPC removes the key E from the state register S.

[a]The $\cdot$ symbol represents scalar multiplication of the bit m with the string c.

Lemma 4.5.11. *Let C be a circuit on W wires that leaves some wire $w \leqslant W$ unmeasured. Let $\mathfrak{I}^C$ be the ideal functionality for C, as described in Definition 4.5.1, and let $\Pi^{\nrightarrow}$ be Protocol 4.5.10 for a computational-basis measurement on w. For all sets $I_\mathcal{A} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_\mathcal{A}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\nrightarrow} \diamond \mathfrak{I}^C)_\mathcal{A})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_\mathcal{S}^{\nrightarrow\circ C})]\right| \leqslant \mathrm{negl}(n).$$

Before we prove Lemma 4.5.11, we prove a separate lemma, capturing the fact that $\mathsf{CNOT}_{1,c}$ makes it hard to alter the outcome of a computational-basis measurement with a (Pauli) attack X^b if b does not depend on c.

The operation $\mathsf{CNOT}_{1,c}$ entangles the data qubit in register M with a random subset of the trap qubits in register T_1, as dictated by c. In Step 4 of Protocol 4.5.10, the MPC checks both for consistency of all the bits entangled by c (they have to match the measured data) *and* all the bits that are not entangled by c (they have to remain zero). Checking the consistency of a measurement outcome after the application of $\mathsf{CNOT}_{1,c}$ is as good as measuring the logical state: any attacker that does not know c will have a hard time influencing the measurement outcome, as he will have to flip all qubits in positions i for which $c_i = 1$ without accidentally flipping any of the qubits in positions i for which $c_i = 0$.

Lemma 4.5.12. *Let $m \in \{0,1\}$, and let ρ be a single-qubit state. Let $p : \{0,1\}^{n+1} \to [0,1]$ be a probability distribution. Then the following expression is upper bounded by 2^{-n}:*

$$\left\| \mathop{\mathbb{E}}_{b,c} \langle m, m\cdot c| \mathsf{X}^b \mathsf{CNOT}_{1,c} \left(\rho \otimes |0^n\rangle\langle 0^n|\right) \mathsf{CNOT}_{1,c}^\dagger \mathsf{X}^b |m, m\cdot c\rangle - p(0^{n+1}) \langle m|\rho|m\rangle \right\|_{\mathrm{tr}},$$

where $b \leftarrow_p \{0,1\}^{n+1}$ and $c \leftarrow_R \{0,1\}^n$.

Proof. By commutation relations between CNOT and X, we have that for all b and c,

$$\mathsf{X}^b \mathsf{CNOT}_{1,c} = \mathsf{CNOT}_{1,c} \mathsf{X}^{b \oplus (0, b_1 \cdot c)}, \tag{4.101}$$

where b_1 denotes the first bit of b. Furthermore, $\mathsf{CNOT}_{1,c}|m, m\cdot c\rangle = |m, 0^n\rangle$. Using these two equalities, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{b \leftarrow_p \{0,1\}^{n+1} \\ c \leftarrow_R \{0,1\}^n}} \langle m, m\cdot c| \mathsf{X}^b \mathsf{CNOT}_{1,c} \left(\rho \otimes |0^n\rangle\langle 0^n|\right) \mathsf{CNOT}_{1,c}^\dagger \mathsf{X}^b |m, m\cdot c\rangle && (4.102)\\
= &\mathop{\mathbb{E}}_{\substack{b \leftarrow_p \{0,1\}^{n+1} \\ c \leftarrow_R \{0,1\}^n}} \left\langle m, 0^n \right| \mathsf{X}^{b\oplus(0,b_1\cdot c)} \left(\rho \otimes |0^n\rangle\langle 0^n|\right) \mathsf{X}^{b\oplus(0,b_1\cdot c)} \left| m, 0^n \right\rangle. && (4.103)
\end{aligned}$$

Let us consider which values of b result in a nonzero term. In order for the last n qubits to be in the $|0^n\rangle\langle 0^n|$ state after $\mathsf{X}^{b\oplus(0,b_1\cdot c)}$, it is necessary that $b \oplus (0, b_1 \cdot c) \in \{(0, 0^n), (1, 0^n)\}$. By considering the two possible cases $b_1 = 0$ and $b_1 = 1$, we see that the only two values of b for which this is the case are $b = (0, 0^n)$ and $b = (1, c)$. Thus Equation (4.103) equals

$$\begin{aligned}
&\mathop{\mathbb{E}}_{c \leftarrow_R \{0,1\}^n} p(0^{n+1}) \left\langle m, 0^n \right| \left(\rho \otimes |0^n\rangle\langle 0^n|\right) \left| m, 0^n \right\rangle \\
&\qquad + p(c) \left\langle m, 0^n \right| \mathsf{X}^{1,0^n} \left(\rho \otimes |0^n\rangle\langle 0^n|\right) \mathsf{X}^{1,0^n} \left| m, 0^n \right\rangle && (4.104)\\
= \quad & p(0^{n+1}) \langle m|\rho|m\rangle + \mathop{\mathbb{E}}_{c \sim \mathcal{U}_n} p(c) \langle m+1|\rho|m+1\rangle && (4.105)\\
\approx_{2^{-n}} \quad & p(0^{n+1}) \langle m|\rho|m\rangle. && (4.106)
\end{aligned}$$

The last step follows from the fact that $\mathbb{E}_c\, p(c) = 2^{-n}$. □

We now move on to proving the security of Protocol 4.5.10 by showing that its outcome resembles that of the ideal functionality.

Proof of Lemma 4.5.11. Let player i be the player holding (the encoding of) the state in wire w (assume, for simplicity, that w is the only wire in the computation). If player i is honest, then it is simple to check that the outcome is correct: the unitary V is designed so that, whatever the first (data) qubit collapses to, all other qubits that appear in s measure to the same value. In step 4, the MPC checks that this is indeed the case, and stores the measured value in the state register.

For the rest of this proof, we will assume that player i is dishonest. The other players do not play a role, except for their power to abort the ideal functionalities and/or MPC. We do not fix which players in $[k] \setminus \{i\}$ are honest: as long as at least one of them is, the encoding key E will be unknown to the adversary.

In an execution of $\Pi^{\nrightarrow} \diamond \mathfrak{I}^C$, an adversary has two opportunities to influence the outcome: before and after interacting with the ideal functionality for C. Before the adversary submits the register $M = R_i^{\mathsf{in}}$ to $\mathfrak{I}^C$, it applies an arbitrary attack unitary A to the register MR it receives from the environment. (Recall that R is a side-information register.) Afterwards, it can act on $MT_1 = R_i^{\mathsf{out}}$ and R, and produces two bits (b_i to signal cheating to $\mathfrak{I}^C$, and b_i' to signal cheating to the MPC which is part of $\Pi^{\nrightarrow}$), plus a bit string. We may assume, without loss of generality, that the adversary first applies the honest unitary V, followed by an arbitrary (unitary) attack B and subsequently by an honest computational-basis measurement of the registers MT_1.

For any adversary, specified by the unitaries A and B, define a simulator $\mathcal{S}$ as follows:

Simulator 4.5.13. On input ρ^{MR} from the environment, do:

1. Run A on registers MR.
2. Sample a random $F \in \mathscr{C}_{n+1}$ and a random $r \in \{0,1\}^{n+1}$.
3. Prepare the state $F|r\rangle\langle r|F^\dagger$ in a separate register XT_1, and apply the map B to XT_1R, using the instruction $F^\dagger$ instead of T.
4. Measure XT_1 in the computational basis, and check that the outcome is r. If so, submit M to $\mathfrak{I}^{\nrightarrow \circ C}$, along with a bit $b = 0$ (no cheating). Otherwise, submit M and $b = 1$.

Throughout this proof, we decompose the attack B as

$$B = \sum_{b,d \in \{0,1\}^{n+1}} \left(\mathsf{X}^b \mathsf{Z}^d\right)^{MT_1} \otimes B_{b,d}^R, \tag{4.107}$$

and similarly as before, we abbreviate $B_b := \sum_d B_{b,d}$ (and B_0 for $B_{0^{n+1}}$).

We analyze the output state in registers RS (note that the MT_1 registers are destroyed by the measurement) in both the ideal and the real case, and aim to show that they are indistinguishable, whatever the input ρ^{MR} was.

In the ideal (simulated) case, first consider the output state in case of accept. Following the steps of the simulator, abbreviating $\sigma = A^{MR}\rho^{MR}A^\dagger$, and decomposing B as in Equation (4.107), we see that the output in RS in case of accept is

$$\sum_{m\in\{0,1\}^m}\mathop{\mathbb{E}}_r \langle m|^M \Phi_C^M\left(\langle r|^{XT_1} B^{XT_1R}\left(\sigma\otimes\left(F^\dagger F|r\rangle\langle r|F^\dagger F\right)^{XT_1}\right)B^\dagger|r\rangle\right)|m\rangle\otimes|m\rangle\langle m|^S \tag{4.108}$$

$$= \sum_{m\in\{0,1\}}\sum_{b,d,b',d'}\mathop{\mathbb{E}}_r \langle m|^M\left(\Phi_C^M\left(B_{b,d}^R\sigma B_{b',d'}^\dagger\right)\otimes\langle r|\mathsf{X}^b\mathsf{Z}^d|r\rangle\langle r|\mathsf{Z}^{d'}\mathsf{X}^{b'}|r\rangle^{XT_1}\right)|m\rangle\otimes|m\rangle\langle m|^S \tag{4.109}$$

$$= \sum_{m\in\{0,1\}}\langle m|^M\Phi_C^M\left(B_0^R A^{MR}\rho^{MR}A^\dagger B_0^\dagger\right)|m\rangle\otimes|m\rangle\langle m|^S. \tag{4.110}$$

The ideal reject case is similar, except we project onto $\mathbb{I}-|r\rangle\langle r|$ instead of onto $|r\rangle\langle r|$. The output state is

$$\begin{aligned}&\sum_{m\in\{0,1\}}\sum_{b\neq 0^{n+1}}\langle m|^M\Phi_C^M\left(B_b^R A^{MR}\rho^{MR}A^\dagger B_b^\dagger\right)|m\rangle\otimes|\bot\rangle\langle\bot|^S\\ &=\sum_{b\neq 0^{n+1}} Tr_M\left[B_b^R A^{MR}\rho^{MR}A^\dagger B_b^\dagger\right]\otimes|\bot\rangle\langle\bot|^S.\end{aligned} \tag{4.111}$$

In the real protocol, the unitary V does not reveal any information about c, so the attack B is independent of it. This allows us to apply Lemma 4.5.12, after performing a Pauli twirl to decompose the attack B. Again abbreviating $\sigma = A\rho A^\dagger$, the state in the accept case is

$$\begin{aligned}=&\mathop{\mathbb{E}}_c\sum_m\langle r\oplus(m,m\cdot c)|^{MT_1}B^{MT_1R}\mathsf{X}^r\mathsf{Z}^s\mathsf{CNOT}_{1,c}E^\dagger E\left(\Phi_C^M(\sigma)\otimes\left|0^n\right\rangle\!\left\langle 0^n\right|^{T_1}\right)\\ &\qquad E^\dagger E\mathsf{CNOT}_{1,c}^\dagger\mathsf{Z}^s\mathsf{X}^r B^\dagger|r\oplus(m,m\cdot c)\rangle\otimes|m\rangle\langle m|^S\end{aligned} \tag{4.112}$$

$$\begin{aligned}=&\mathop{\mathbb{E}}_c\sum_{m,b}\langle m,m\cdot c|\mathsf{X}^b\mathsf{CNOT}_{1,c}\left(\Phi_C^M\left(B_b^R\sigma B_b^\dagger\right)\otimes\left|0^n\right\rangle\!\left\langle 0^n\right|\right)\\ &\qquad\mathsf{CNOT}_{1,c}^\dagger\mathsf{X}^b|m,m\cdot c\rangle\otimes|m\rangle\langle m|^S\end{aligned} \tag{4.113}$$

$$\approx_{2^{-n}} \text{Eq. (4.110)}. \tag{4.114}$$

For the last step, observe that the probabilities $p(b)$ in the statement of Lemma 4.5.12 are part of B_b.

Similarly, the real reject state is

$$\mathbb{E}_{c} \sum_{m,b} \sum_{x \neq (m, m \cdot c)} \langle x|^{MT_1} \mathsf{X}^b \mathsf{CNOT}_{1,c} \left(\Phi_C^M \left(B_b^R \sigma B_b^\dagger \right) \otimes |0^n\rangle\langle 0^n|^{T_1} \right) \mathsf{CNOT}_{1,c}^\dagger \mathsf{X}^b |x\rangle \otimes |\bot\rangle\langle\bot|^S \tag{4.115}$$

$$\approx_{2^{-n}} \text{Eq. (4.111).} \tag{4.116}$$

In summary, we have shown that the output state in the real case is close to Eq. (4.110) + Eq. (4.111), for any input state ρ^{MR} provided by the environment $\mathcal{E}$. □

4.5.4 Subprotocol: decoding

After the players run the computation subprotocols for all gates in the Clifford and measurement circuit, all they need to do is to decode their wires to recover their output. At this point, there is no need to check the authentication traps publicly: there is nothing to gain for a dishonest player by incorrectly measuring or lying about their measurement outcome. Hence, it is sufficient for all (honest) players to apply the regular decoding procedure for the Clifford code.

Below, we describe the decoding procedure for a single wire held by one of the players. If there are multiple output wires, then Protocol 4.5.14 can be run in parallel for all those wires.

Protocol 4.5.14 (Decoding)**.** Player i holds an encoding of the state w in the register MT_1. The classical MPC holds the encoding key E in the state register S.

1. MPC sends E to player i, removing it from the state register S.
2. Player i applies E to register MT_1.
3. Player i measures T_1 in the computational basis. If the outcome is not 0^n, player i discards M and aborts the protocol.

Lemma 4.5.15. *Let C be a circuit on W wires that leaves a single wire $w \leqslant W$ (intended for player i) unmeasured. Let $\mathfrak{I}^C$ be the ideal functionality for C, as described in Definition 4.5.1, and let $\mathfrak{I}_C^{\mathsf{MPQC}}$ be the ideal MPQC functionality for C, as described in Definition 4.2.2. Let Π^{Dec} be Protocol 4.5.14 for decoding wire w. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\mathsf{Dec}} \diamond \mathfrak{I}^C)_{\mathcal{A}})] = \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{C,\mathcal{S}}^{\mathsf{MPQC}})].$$

Proof sketch. If player i is honest, then he correctly decodes the state received from the ideal functionality $\mathfrak{I}^C$. A simulator would only have to compute the adversary's abort bit for $\mathfrak{I}_C^{\mathsf{MPQC}}$ based on whether the adversary decides to abort in either $\mathfrak{I}^C$ or the MPC computation in Π^{Dec}.

If player i is dishonest, a simulator $\mathcal{S}$ runs the adversary on the input state received from the environment before inputting the resulting state into the ideal functionality $\mathfrak{I}_C^{\mathsf{MPQC}}$. The simulator then samples a key for the Clifford code and encodes the output of $\mathfrak{I}_C^{\mathsf{MPQC}}$, before handing it back to the adversary. It then simulates Π^{Dec} by handing the sampled key to the adversary. If the adversary aborts in one of the two simulated protocols, then the simulator sends abort to the ideal functionality $\mathfrak{I}_C^{\mathsf{MPQC}}$. □

4.5.5 Combining subprotocols

Finally, we show how to combine the subprotocols from the previous sections in order to perform multi-party Clifford computation.

Recalling the notation from Definition 4.2.2, let C be a quantum circuit on $W \in \mathbb{N}_{>0}$ wires, which are partitioned into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. We assume that C is decomposed in a sequence $G_1, \ldots, G_m$ of operations where each G_i is one of the following operations:

- a single-qubit Clifford on some wire $j \in [M]$, possibly controlled on a classical value;
- a CNOT on wires $j_1, j_2 \in [M]$ for $j_1 \neq j_2$, possibly controlled on a classical value;
- a measurement of the qubit on wire j in the computational basis.

In Sections 4.4 and 4.5.1 to 4.5.3, we have presented subprotocols for encoding single qubits and perform these types of operations on single wires. The protocol for all players to jointly perform the bigger computation C is simply a concatenation of those smaller subprotocols:

Protocol 4.5.16 (Encoding and Clifford+measurement computation)**.** Let C be a Clifford + measurement circuit composed of the gates $G_1, \ldots, G_m$ on wires $[W]$ as described above.

1. For all $i \in [k]$ and $j \in R_i^{\mathsf{in}}$, run Protocol 4.4.2 for the qubit in wire j to encode the inputs.

2. For all $j \in R^{\mathsf{ancilla}}$, run Protocol 4.4.2 (with the differences described in Section 4.4.2) to encode the ancillary qubits.
3. For all $j \in [m]$:
   (a) If G_j is a single-qubit Clifford, run Protocol 4.5.2 for G_j.
   (b) If G_j is a CNOT, run Protocol 4.5.5 for G_j.
   (c) If G_j is a computational-basis measurement, run Protocol 4.5.10 for G_j.
4. For all $i \in [k]$ and $j \in R_i^{\mathsf{out}}$, run Protocol 4.5.14 for the qubit in wire j to decode the outputs.

Lemma 4.5.17. *Let Π^{Cliff} be Protocol 4.5.16, and $\mathfrak{I}^{\mathsf{Cliff}}$ be the ideal functionality described in Definition 4.2.2 for the special case where the circuit consists of (a polynomial number of) Cliffords and measurements. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{\mathsf{Cliff}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{\mathsf{Cliff}})| \leqslant \mathrm{negl}(n).$$

Proof. We start by introducing some notation. For a circuit C composed of gates $G_1, \ldots, G_m$, we write $\Pi^C := \Pi^{G_m} \diamond \cdots \diamond \Pi^{G_1}$. Also, we write $C_{i,j} := G_j \circ \cdots \circ G_i$ for the subcircuit from gates i through j (for $1 \leqslant i \leqslant j \leqslant m$), so that $\Pi^C = \Pi^{C_{1,m}}$. For the real protocols, we will omit the subscript $\mathcal{A}$, implicitly breaking up the adversary into multiple algorithms that attack each subprotocol.

Note that the real protocol $\Pi^{\mathsf{Cliff}} = \Pi^{\mathsf{Dec}} \diamond \Pi^C \diamond \Pi^{\mathsf{Enc}}$. Initially, we compare the real protocol *without decoding* with the functionality for ideal quantum k-party computation without decoding given in Definition 4.5.1. More specifically, we will prove by induction that for all i, there exists a simulator $\mathcal{S}'$ such that following holds holds for all environments $\mathcal{E}'$:

$$\left|\Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^C \diamond \Pi^{\mathsf{Enc}})\right] - \Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}_{\mathcal{S}'}^{C_{1,i-1}})\right]\right| \leqslant i \cdot \mathrm{negl}(n), \tag{4.117}$$

where $\mathfrak{I}^{C_{1,i-1}}$ refers to the ideal functionality without decoding described in Definition 4.5.1: recall that it returns *encoded* states to each player.

For the basis case $i = 1$ (yielding the empty circuit $C_{1,0}$), Lemma 4.4.3 guarantees the existence of a simulator $\mathcal{S}'$ such that for all $\mathcal{E}'$,

$$\left|\Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{\mathsf{Enc}})\right] - \Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \mathfrak{I}_{\mathcal{S}'}^{\mathsf{Enc}})\right]\right| \leqslant \mathrm{negl}(n), \tag{4.118}$$

where $\mathfrak{I}^{\mathsf{Enc}}$ is the ideal functionality of the encoding given in Definition 4.4.1. In particular, for every $\mathcal{E}''$ we have that

$$\left|\Pr\left[1 \leftarrow (\mathcal{E}'' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}})\right] - \Pr\left[1 \leftarrow (\mathcal{E}'' \leftrightarrows \Pi^{C} \diamond \mathfrak{I}^{\mathsf{Enc}}_{\mathcal{S}'})\right]\right| \leqslant \mathrm{negl}(n). \tag{4.119}$$

For the induction step, assume that our statement holds for some $i \geqslant 1$. If G_{i+1} is a single-qubit Clifford, then there exist simulators $\mathcal{S}'$ and $\mathcal{S}''$ such that

$$\begin{aligned}
&\left|\Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}})\right] - \Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i+1,m}} \diamond \mathfrak{I}^{C_{1,i}}_{\mathcal{S}'})\right]\right| && (4.120)\\
&\leqslant \left|\Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}})\right] - \Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}^{C_{1,i-1}}_{\mathcal{S}''})\right]\right| \\
&\qquad + \left|\Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}^{C_{1,i-1}}_{\mathcal{S}''})\right] - \Pr\left[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i+1,m}} \diamond \mathfrak{I}^{C_{1,i}}_{\mathcal{S}'})\right]\right| && (4.121)\\
&\leqslant (i+1)\,\mathrm{negl}(n). && (4.122)
\end{aligned}$$

In the first step, we used the triangle inequality. In the second step, we used the induction hypothesis and Lemma 4.5.3. In case G_{i+1} is a CNOT or measurement, the same argument follows by using Lemmas 4.5.6 and 4.5.11 accordingly.

Finally, by Lemma 4.5.15, we can also replace $\Pi^{\mathsf{Dec}} \diamond \mathfrak{I}^{C}$ by $\mathfrak{I}^{\mathsf{Cliff}}$ (which returns a plaintext output to the players), at the cost of $\mathrm{negl}(n)$. We note that the $m = \mathrm{poly}(n)$ negligible functions we accumulated by the use of Lemmas 4.5.3, 4.5.6, 4.5.11 and 4.5.15 only depend on the type of operation and not on the position i. Therefore, the result follows since $m \cdot \mathrm{negl}(n) = \mathrm{negl}(n)$. □

4.6 Protocol: general quantum circuits

In this section, we show how to lift the MPQC protocol for Clifford operations (as laid out in Sections 4.4 and 4.5) to MPQC for general quantum circuits.

The main idea is to use magic states for T gates, as described in Sections 2.4.4 and 3.4.2. Our main difficulty here is that the magic states must be supplied by the possibly dishonest players themselves. We solve this problem in two steps: first, player 1 is asked to produce a large number of encodings of (supposed) T magic states, and the other players will test a fraction of them (see Section 4.6.1). After this step, if none of the players have reported an error during the testing, then with high probability the remaining resource states are at least somewhat good. As the second step, the players run a distillation procedure to further increase the quality of the magic states (see Section 4.6.2).

The protocols presented in Sections 4.6.1 and 4.6.2 describe how to distill magic states from a list of untrusted states in the plaintext setting. The distillation procedure consists entirely of (classically controlled) Clifford operations and computational-basis measurements. Thus, we can execute the entire distillation using the protocols from Sections 4.4 and 4.5. For details on the final multi-party quantum computation protocol, see Section 4.6.3.

4.6.1 Step 1: sampling

Classically, some properties of a bit string can be estimated just by querying a small random fraction of it. For instance, in order to estimate the Hamming weight of an n-bit string x, one could calculate the Hamming weight w_s of a random sample of ℓ of the bits of x, and get the guarantee that $|x| \in \left[\frac{nw_s}{\ell} - \delta, \frac{nw_s}{\ell} + \delta\right]$ except with probability $O(2^{-\delta^2 \ell})$ [Hoe63; BF10].

Such a statement does not trivially carry over to the quantum setting, since the tested quantum state could, for instance, be entangled with the environment. However, Bouman and Fehr [BF10], having studied this problem in the quantum setting, showed that such sampling arguments *are* possible, albeit with a quadratic loss in the error probability. A corollary of their result that will be important in this work is the following.

Lemma 4.6.1 (Application of [BF10, Theorem 3]). *Let* $|\varphi_{AE}\rangle \in (\mathbb{C}^2)^{\otimes n} \otimes \mathcal{H}_E$ *be a quantum state and let* $B = \{|v_0\rangle, |v_1\rangle\}$ *be a fixed single-qubit basis. If we measure* ℓ *random qubits of* $Tr_E\left(|\varphi_{AE}\rangle\langle\varphi_{AE}|\right)$ *in the* B*-basis and all of the outcomes are* $|v_0\rangle$*, then with probability* $1 - O(2^{-\delta^2 \ell})$*, we have that*

$$|\varphi_{AE}\rangle \in \operatorname{span}\left(\pi(|v_0\rangle^{\otimes n-t} \otimes |v_1\rangle^{\otimes t}) \otimes |\psi\rangle \;\middle|\; 0 \leqslant t \leqslant \delta n, \pi \in S_n, |\psi\rangle \in \mathcal{H}_E\right).$$

In words, if ℓ out of n qubits are tested and all turn out to be in the "correct" state $|v_0\rangle$, then with high probability the original list of qubits had most of its weight (except for a fraction up to δn) on that state. In Section 4.6.3, we will use this fact when all players test a fraction of the magic states created by player 1.

4.6.2 Step 2: distillation

Bravyi and Kitaev [BK05] proposed a distillation protocol that allows the creation of states that are δ-close to true magic states, given poly(log(1/δ)) copies of *noisy* magic states. Specifically, writing $|\mathsf{T}^{\perp}\rangle := \mathsf{T}|-\rangle$, we have:

Theorem 4.6.2 (Magic-state distillation [BK05]). *There exists a circuit of* CNOT*-depth* $d_{\text{distill}}(n) \leqslant O(\log(n))$ *consisting of* $p_{\text{distill}}(n) \leqslant \text{poly}(n)$ *many classically controlled Cliffords and computational-basis measurements such that for any* $\varepsilon < \frac{1}{2}\left(1 - \sqrt{3/7}\right) \approx 0.173$*, if* ρ *is the output on the first wire when the circuit is run on input*

$$\left((1-\varepsilon)|\mathsf{T}\rangle\langle\mathsf{T}| + \varepsilon|\mathsf{T}^{\perp}\rangle\langle\mathsf{T}^{\perp}|\right)^{\otimes n}, \tag{4.123}$$

then $1 - \langle\mathsf{T}|\rho|\mathsf{T}\rangle \leqslant O\left((5\varepsilon)^{n^c}\right)$*, where* $c = (\log_2 30)^{-1} \approx 0.2$.

Theorem 4.6.2 requires that the inputs to the distillation circuit are already ε-close to the correct state $|\mathsf{T}\rangle$. Our starting point is a bit different, since our MPQC protocol asks a possibly dishonest player to prepare the states. We show that the output of the sampling step from Section 4.6.1 suffices as an input to the magic-state distillation protocol, as long as we properly dephase and permute it. The full circuit (for creating t magic states from m qubits that remain after sampling) is as follows.

Circuit 4.6.3 (Magic-state distillation)**.** Given an m-qubit input state and a parameter $t < m$:

1. To each qubit, apply $\hat{\mathsf{Z}} := \mathsf{PX}$ with probability $1/2$.
2. Permute the qubits by a random $\pi \in S_m$.
3. Divide the m qubits into t blocks of size m/t, and apply magic-state distillation from Theorem 4.6.2 to each block.

Remark. Circuit 4.6.3 can be implemented with (classically controlled) Clifford gates and measurements in the computational basis.

For the rest of this section, fix a basis $|\hat{0}\rangle := |\mathsf{T}\rangle$ and $|\hat{1}\rangle := |\mathsf{T}^{\perp}\rangle$. For strings $w \in \{0,1\}^m$, we will write $|\hat{w}\rangle := |\hat{w}_1\rangle \otimes \cdots \otimes |\hat{w}_m\rangle$. In this basis, the all-zero string of length m represents m copies of $|\mathsf{T}\rangle$.

We analyze Circuit 4.6.3. Note that $\hat{\mathsf{Z}} = |\hat{0}\rangle\langle\hat{0}| - |\hat{1}\rangle\langle\hat{1}|$ (up to a global phase). The first step of the circuit is to apply $\hat{\mathsf{Z}}$ with probability $1/2$ to each qubit, which has the effect of *dephasing* the qubit, or equivalently, making the state diagonal, in the $\{|\hat{0}\rangle, |\hat{1}\rangle\}$ basis. More precisely, if we let $\rho = \sum_{w,w' \in \{0,1\}^m} \alpha_{w,w'} |\hat{w}\rangle\langle\hat{w}'|$, applying $\hat{\mathsf{Z}}$ with probability $1/2$ has the following effect:

$$\rho \mapsto \sum_{w,w' \in \{0,1\}^m} \alpha_{w,w'} \bigotimes_{i=1}^{m} \frac{1}{2}\left(|\hat{w}_i\rangle\langle\hat{w}'_i| + \hat{\mathsf{Z}}|\hat{w}_i\rangle\langle\hat{w}'_i|\hat{\mathsf{Z}}\right) \tag{4.124}$$

$$= \sum_{w,w' \in \{0,1\}^m} \alpha_{w,w'} \bigotimes_{i=1}^{m} \frac{1}{2}\left(|\hat{w}_i\rangle\langle\hat{w}'_i| + (-1)^{w_i + w'_i}|\hat{w}_i\rangle\langle\hat{w}'_i|\right) \tag{4.125}$$

$$= \sum_{w \in \{0,1\}^m} \alpha_{w,w} |\hat{w}\rangle\langle\hat{w}| =: \rho'. \tag{4.126}$$

Let Ξ' denote the quantum channel given by steps 2–3 of Circuit 4.6.3. Note that Ξ' is symmetric: any inputs ρ_1 and $\rho_2 = \pi\rho_1\pi^\dagger$ will both be mapped to $\frac{1}{m!}\sum_{\pi' \in S_m} \pi'\rho_1\pi'^\dagger$ after step 2. Thus, the following theorem applies, where for any $\ell \leqslant m$, Π_ℓ is the orthogonal projector onto $\mathrm{span}\{\pi(|\mathsf{T}\rangle^{\otimes m-w}|\mathsf{T}^\perp\rangle^{w}) \mid w \leqslant \ell, \pi \in S_m\}$.

Theorem 4.6.4 ([DNS12, Theorem D.1])**.** *Let $\ell \leqslant m$, let σ be an m-qubit state, diagonal in the basis $\{|\hat{w}\rangle : w \in \{0,1\}^m\}$, and suppose $\Pi_\ell\sigma = \sigma$. Let Ξ' be any CPTP map from m qubits to t qubits such that $\Xi'(\pi\omega\pi^\dagger) = \Xi'(\omega)$ for any n-qubit state ω and any $\pi \in S_m$.*

Then, letting $\delta_s = \frac{s}{m}$*:*

$$\begin{aligned}&\left\| \Xi'(\sigma) - \left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t} \right\|_{\mathrm{tr}} \\ \leqslant (m+1)\max_{s\leqslant\ell} &\left\| \Xi'\left(\left((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s |\hat{1}\rangle\langle\hat{1}|\right)^{\otimes m}\right) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \right\|_{\mathrm{tr}}.\end{aligned}$$

The above observations allow us to prove the following lemma, which states that the output of Circuit 4.6.3 is exponentially (in m/t) close to the desired state (t copies of the magic state $|\mathsf{T}\rangle$), if we start with a state on which the sampling procedure was successful.

Lemma 4.6.5. *Let* Ξ *denote the CPTP map induced by Circuit 4.6.3. If* ρ *is an* m*-qubit state such that* $\mathrm{Tr}(\Pi_\ell \rho) \geqslant 1-\varepsilon$*, then*

$$\| \Xi(\rho) - (|\mathsf{T}\rangle\langle\mathsf{T}|)^{\otimes t} \|_{\mathrm{tr}} \leqslant O\left(m\sqrt{t}\left(\frac{\ell}{m}\right)^{O((m/t)^c/2)} + \varepsilon \right),$$

for some constant $c > 0$.

Proof. Let ρ' be as in Equation (4.126). We have

$$\mathrm{Tr}(\Pi_\ell \rho') = \mathrm{Tr}(\Pi_\ell \rho) \geqslant 1-\varepsilon. \tag{4.127}$$

Thus, we can write $\rho' = (1-\varepsilon)\sigma + \varepsilon\sigma'$ for $\sigma = \frac{1}{1-\varepsilon}\Pi_\ell\rho'$, and σ' another quantum state. We will first bound the trace distance between $\Xi'(\sigma)$ and the desired state $\left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t}$, and later on will use a trivial upper bound of 1 for the trace distance between $\Xi'(\sigma')$ and that desired state.

On a symmetric state, Ξ' is simply the state distillation protocol of Bravyi and Kitaev [BK05], applied t times in parallel to m/t qubits each time. Let Φ be one state distillation protocol distilling one qubit from m/t (so Ξ' acts as $\Phi^{\otimes t}$ on symmetric states). Abbreviating

$$\mu_s := \left((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s|\hat{1}\rangle\langle\hat{1}|\right)^{\otimes m/t}, \tag{4.128}$$

and applying Theorem 4.6.4, we get

$$\left\| \Xi'(\sigma) - \left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t} \right\|_{\mathrm{tr}} \tag{4.129}$$

$$\leqslant (m+1)\max_{s\leqslant\ell} \left\| \Xi'\left(\left((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s|\hat{1}\rangle\langle\hat{1}|\right)^{\otimes m}\right) - \left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t} \right\|_{\mathrm{tr}} \tag{4.130}$$

$$= (m+1)\max_{s\leqslant\ell} \left\| \Phi(\mu_s)^{\otimes t} - \left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t} \right\|_{\mathrm{tr}} \tag{4.131}$$

$$\leqslant 2(m+1)\max_{s\leqslant\ell}\sqrt{1 - (\langle\hat{0}|\Phi(\mu_s)|\hat{0}\rangle)^t}. \tag{4.132}$$

The last step is due to the general inequality $\| \rho - |\psi\rangle\langle\psi| \|_{\mathrm{tr}} \leqslant 2\sqrt{1-\langle\psi|\rho|\psi\rangle}$ that relates trace distance to fidelity.

By Theorem 4.6.2, using $\delta_s = \frac{s}{m} \leqslant \frac{\ell}{m}$, we have

$$1-\langle\hat{0}|\,\Phi(\mu_s)\,|\hat{0}\rangle \leqslant O\Big((5\delta_s)^{(m/t)^c}\Big) \leqslant O\left(\left(5\frac{\ell}{m}\right)^{(m/t)^c}\right) \tag{4.133}$$

for $c \approx 0.2$. Now abbreviate $\delta := (5\ell/m)^{(m/t)^c}$. Since the fidelity is bounded between 0 and 1, it follows that

$$1-\big(\langle\hat{0}|\,\Phi(\mu_s)\,|\hat{0}\rangle\big)^t \leqslant O(1-(1-\delta)^t). \tag{4.134}$$

Continuing our derivation, we have (using the general inequalities for $x \leqslant \frac{1}{2}$, that state $1-x \geqslant e^{-2x} \geqslant 1-2x$)

$$\begin{aligned}
\text{Eq. (4.132)} &\leqslant O\Big(2(m+1)\sqrt{1-(1-\delta)^t}\Big) && (4.135)\\
&\leqslant O\Big(2(m+1)\sqrt{1-e^{-2\delta t}}\Big) && (4.136)\\
&\leqslant O\Big(2(m+1)\sqrt{2\delta t}\Big). && (4.137)\\
& && (4.138)
\end{aligned}$$

That concludes our upper bound on the trace distance between $\Xi'(\sigma)$ and $\big(|\hat{0}\rangle\langle\hat{0}|\big)^{\otimes t}$. Putting everything together, we have

$$\begin{aligned}
\Big\| \Xi(\rho) - \big(|\hat{0}\rangle\langle\hat{0}|\big)^{\otimes t} \Big\|_{\mathrm{tr}} &= \Big\| \Xi'(\rho') - \big(|\hat{0}\rangle\langle\hat{0}|\big)^{\otimes t} \Big\|_{\mathrm{tr}} && (4.139)\\
&= \| (1-\varepsilon)\Xi'(\sigma) + \varepsilon\Xi'(\sigma') - \big(|\hat{0}\rangle\langle\hat{0}|\big)^{\otimes t} \|_{\mathrm{tr}} && (4.140)\\
&\leqslant (1-\varepsilon)O\Big(2(m+1)\sqrt{2\delta t}\Big) + \varepsilon = O\Big(m\sqrt{t}(5\ell/m)^{(m/t)^c/2} + \varepsilon\Big). && (4.141)
\end{aligned}$$

□

4.6.3 Protocol: MPQC for general circuits

In this section, we describe how to incorporate the sampling and distillation techniques into a protocol for multi-party quantum computation of circuits that may contain (classically controlled) Clifford gates, computational-basis measurements, *and* T gates. Essentially, we lift the procedures described in Section 4.6.1 and Circuit 4.6.3 into the authenticated setting.

Protocol 4.6.6 (Magic-state creation)**.** Let t be the number of magic states we wish to create. Let $\ell := (t+k)n$.

1. Player 1 creates ℓ copies of $|\mathsf{T}\rangle$ and encodes them separately using Protocol 4.4.2 (jointly with the other players).
2. MPC picks random disjoint sets $S_2, \ldots, S_k \subseteq [\ell]$ of size n each.
3. For each $i \in 2, \ldots, k$, player i decodes the magic states indicated by S_i (see Protocol 4.5.14), measures in the $\{|\mathsf{T}\rangle, |\mathsf{T}^{\perp}\rangle\}$-basis and aborts if any outcome is different from $|\mathsf{T}\rangle$.
4. On the remaining encoded states, the players run Protocol 4.5.16 for multi-party computation of Clifford circuits (but skipping the input-encoding and output-decoding steps) to perform the magic-state distillation protocol described in Circuit 4.6.3. Any randomness required in that protocol is sampled by the classical MPC.

Protocol 4.6.6 implements the following ideal functionality for creating t magic states, up to a negligible error:

Definition 4.6.7 (Ideal functionality for magic-state creation)**.** Let t be the number of magic states we wish to create. Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.

1. For every $i \in I_{\mathcal{A}}$, player i sends a bit b_i to the trusted third party.
   (a) If $b_i = 0$ for all i, the trusted third party samples t random $(n+1)$-qubit Clifford E_j for $1 \leqslant j \leqslant t$, and sends $E_j(|\mathsf{T}\rangle \otimes |0^n\rangle)$ to Player 1.
   (b) Otherwise, the trusted third party sends `abort` to all players.
2. Store the keys E_j, for $1 \leqslant j \leqslant t$ in the state register S of the ideal functionality.

Lemma 4.6.8. *Let Π^{MS} be Protocol 4.6.6, and $\mathfrak{I}^{MS}$ be the ideal functionality described in Definition 4.6.7. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{MS})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{MS})\right| \leqslant \mathrm{negl}(n).$$

Proof. The simulator for Π^{MS} is similar to the composed simulator for $\Pi^{\mathsf{Dec}} \diamond \Pi^{C} \diamond \Pi^{\mathsf{Enc}}$, where C is Circuit 4.6.3. The difference is that the input is now chosen by player 1 instead of being given by the environment, and that each player tests if the decoded qubit is correct. We make a small modification for each of the following cases:

Case 1: player 1 is honest. In this case, the simulator only needs to also set $b_i = 1$ whenever the adversary aborts after it receives the output of the ideal quantum computation in Step 3 of Protocol 4.6.6. Otherwise, the simulator is exactly the same as the composed one.

Case 2: player 1 is dishonest. In this case, the simulator runs the entire Protocol 4.6.6, simulating all players, including the honest ones. If any of the honest players aborts in Step 3, the simulator aborts by setting $b_1 = 1$. For the other adversaries, the simulator sets $b_i = 1$ if that adversary aborts in this step, similarly to the previous case. Notice that the simulator aborts at this step with the same probability that any of the players would abort in Step 3 of the real protocol.

If none of the players (honest or dishonest) aborts, the simulator replaces the qubits in $[\ell] \setminus (\bigcup_{2 \leqslant i \leqslant k} S_i)$ by honestly generated magic states $|\mathsf{T}\rangle$, which it encodes under the same keys given before by the (simulated) MPC, and continues the composed simulation

We now argue that when there is no abort, the output of Π^{MS} is exponentially close to that of $\mathfrak{I}^{MS}$. Notice that picking the disjoint $S_2, \dots, S_k \subseteq [\ell]$ uniformly at random is equivalent to first picking the test sets $\{S_i\}_{i \in I_{\mathcal{A}}}$ from $[\ell]$, and then picking $\{S_i\}_{i \notin \mathcal{A}}$ from the remaining $[\ell] \setminus (\bigcup_{i \in \mathcal{A}} S_i)$ elements. From this perspective, if the honest players do not abort in Step 3, then Lemma 4.6.1 implies that the state created by player 1 in the other positions $[\ell] \setminus \left(\bigcup_{i \notin \mathcal{A}} S_i\right)$ is $O(2^{\varepsilon^2(k-|I_{\mathcal{A}}|)n})$-close to the the subspace $\mathrm{span}\left(\pi\left(|\mathsf{T}\rangle^{\otimes tn-j}\left|\mathsf{T}^{\perp}\right\rangle^{\otimes j}\right) \middle| 0 \leqslant j \leqslant \varepsilon tn, \pi \in S_{tn}\right)$. If we choose $\varepsilon \leqslant \frac{1}{2}\left(1 - \sqrt{3/7}\right)$, by Lemma 4.6.5 and the union bound, the output of the distillation procedure is $O(t\varepsilon)^{n^c}$-close to $|\mathsf{T}\rangle^{\otimes t}$. In this case, the output of Π^{MS} will be $\mathrm{negl}(n)$-close to encodings of $|T\rangle^{\otimes t}$, which is the output of $\mathfrak{I}^{MS}$ in the no-abort case. □

Given Protocol 4.6.6 for generating encoded magic states, performing universal quantum computation in the multi-party setting is a matter of chaining the protocols together.

For this setting, we consider quantum circuits $C = G_m \cdots G_1$ where G_i can be single-qubit Cliffords, CNOTs, measurements or, additionally, T gates. We will consider a circuit C' where each gate $G_i = \mathsf{T}$ acting on qubit j is replaced by the a magic T-gate computation as depicted in Figure 2.1, acting on the qubit j and a fresh new T magic state.

Protocol 4.6.9 (Protocol for universal MPQC)**.** Let C be a polynomial-sized quantum circuit, and t be the number of T-gates in C.

1. Run Protocol 4.6.6 to create t magic states.

2. Run Protocol 4.5.16 for the circuit C', which is equal to the circuit C, except each T gate is replaced with the circuit from Figure 2.1.

Theorem 4.6.10. *Let Π^{MPQC} be Protocol 4.6.9, and $\mathfrak{I}^{\mathsf{MPQC}}$ be the ideal functionality described in Definition 4.2.2. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr\left[1 \leftarrow \left(\mathcal{E} \leftrightarrows \Pi^{\mathsf{MPQC}}_{\mathcal{A}}\right)\right] - \Pr\left[1 \leftarrow \left(\mathcal{E} \leftrightarrows \mathfrak{I}^{\mathsf{MPQC}}_{\mathcal{S}}\right)\right]\right| \leqslant \mathrm{negl}(n).$$

Proof. Direct from Lemmas 4.5.17 and 4.6.8. □

4.7 Conclusion

In this chapter, we constructed a k-party quantum computation protocol that is computationally secure against $k-1$ actively colluding adversaries. We have shown its security in the real-vs.-ideal paradigm, adopting an unconventional "inductive" approach: rather than showing our subprotocols to fulfill certain ideal subfunctionalities, and then chaining those together, we showed that a smaller ideal subfunctionality can be combined with a subprotocol to achieve a slightly bigger ideal subfunctionality. Repeating this step, we eventually arrived at the ideal functionality $\mathfrak{I}^{\mathsf{MPQC}}$. The reason for taking this inductive approach is that our subprotocols crucially ensure that they output properly *authenticated* states. In the ideal realm, the concept of authentication is not present, so it is not possible to capture this crucial property of our subprotocols by ideal functionalities only.

For completeness, we give an overview of our protocol's (quantum) round complexity, and the number of calls to the classical MPC.

Protocol 4.4.2 encodes a single-qubit input (or an ancilla $|0\rangle$ state) using k rounds of quantum communication and $O(1)$ MPC calls. Note that this protocol can be run in parallel for all input qubits per player, simultaneously for all players. Hence, the overall number of communication rounds for the encoding phase remains k, and the total number of calls to the MPC is $O(w)$ where w is the total number of qubits.

Protocol 4.5.2 for single-qubit Cliffords, Protocol 4.5.10 for measuring in the computational basis and Protocol 4.5.14 for decoding do not require quantum communication and use $O(1)$ MPC calls each, whereas Protocol 4.5.5 for CNOT requires at most $k+2$ rounds of quantum communication, and makes $O(1)$ MPC calls. Overall, Protocol 4.5.16 for encoding and Clifford+measurement computation requires $O(dk)$ rounds of quantum communication and $O(w+g)$ calls to the MPC, where d is the CNOT-depth of the quantum circuit, and g is the total number of gates in the circuit.

Protocol 4.6.6 for magic-state creation encodes $\ell := (t+k)n$ qubits in parallel using k rounds of quantum communication (which can be done in parallel with the actual input encoding, and therefore does not require any extra communication rounds), and $O((t+k)n)$ MPC calls. Then a circuit of size $p_{\text{distill}}(n)$ and CNOT-depth $d_{\text{distill}}(n)$ classically controlled Cliffords and measurements is run on each of the t blocks of n qubits each, which can be done in parallel for the t blocks, requiring $O(k \cdot d_{\text{distill}}(n))$ rounds of quantum communication and $O(tn \cdot p_{\text{distill}}(n))$ calls to the MPC.

Eventually, all T-gate operations in the original circuit C are replaced by the T-gadget from Figure 2.1, resulting in one CNOT and classically controlled Cliffords. Overall, our Protocol 4.6.9 for universal MPQC requires $O(k \cdot (d_{\text{distill}}(n) + d))$ rounds of quantum communication and $O(tn \cdot p_{\text{distill}}(n) + w + g)$ calls to the classical MPC, where d is the $\{\mathsf{CNOT}, \mathsf{T}\}$-depth of the circuit, w is the total number of qubits and g is the total number of gates in the circuit.

We notice that instead of evaluating each Clifford operation gate-by-gate, we could evaluate a general w-qubit Clifford using $O(k)$ rounds of quantum communication, similarly to the CNOT protocol. This could improve the parameter d to be the T depth of the circuit, at the cost of requiring the players to communicate significantly larger states per round.

4.7.1 Future directions

Our results leave a number of exciting open problems to be addressed in future work. The first class of open problems concerns applications of MPQC. For instance, classical MPC can be used to devise zero-knowledge proofs [IKOS09] and digital signature schemes [CDG+17].

An interesting open question concerning our protocol more specifically is whether the CNOT subprotocol can be replaced by a different one that has round complexity independent of the total number of players, reducing the quantum round complexity of the whole protocol. We see no fundamental reason for testing the authenticity of the encoded states multiple times during the protocol, rather than only at decoding time: in principle, any evidence of cheating that happens during the protocol should be carried over to decoding via the traps. Without the intermediate tests, however, we do not know how to prove the security of our protocol against active adversaries as we do now.

We also wonder if it is possible to develop more efficient protocols for narrower classes of quantum computation, instead of arbitrary (polynomial-size) quantum circuits.

Finally, it is interesting to investigate whether the public authentication test we use can be leveraged in quantum protocols for specific MPC-related tasks like oblivious transfer.

5 Quantum homomorphic encryption for Clifford circuits

Chapter contents

5.1 Introduction

The 2009 discovery of fully homomorphic encryption (FHE) in classical cryptography is widely considered to be one of the major breakthroughs of the field [Gen09]. Unlike standard encryption, FHE enables noninteractive computation on encrypted data even by parties that do not hold the decryption key. Crucially, the input, output, and all intermediate states of the computation remain encrypted, and thus hidden from the computing party. While FHE has some obvious applications (e.g., cloud computing), its importance in cryptography stems from its wide-ranging applications to other cryptographic scenarios. For instance, FHE can be used to construct secure two-party computation, efficient zero-knowledge proofs for NP, functional encryption [GKP+13a; GVW13; GKP+13b; SW14], and indistinguishability obfuscation [BB12; GGH+13]. In fact, the breadth of its usefulness has led some to dub FHE “the swiss army knife of cryptography” [BB12].

Early classical FHE schemes were limited in the sense that they could not facilitate arbitrary operations on the encrypted data: some early schemes only implemented a single operation (addition or multiplication) [RSA78; GM84; Pai01]; later on it became possible to combine several operations in a limited way [BGN05; GHV10; SYY99]. Gentry’s first fully-homomorphic-encryption scheme [Gen09] relied on several non-standard computational assumptions. Subsequent work [BV14; BGV12; GW13] has relaxed these assumptions or replaced them with more conventional assumptions such as the hardness of the learning-with-errors (LWE) problem, which is believed to be hard also for quantum attackers. It is impossible to completely get rid of computational assumptions for a classical FHE scheme, since the existence of such a scheme would imply the existence of an information-theoretically secure protocol for private information retrieval (PIR) [KO97] that breaks the lower bound on the amount of communication required for that task [CKGS98; Fil12].

Recent progress on constructing quantum computers has led to theoretical research on “cloud-based” quantum computing. In such a setting, it is natural to ask whether users can keep their data secret from the server that performs the quantum computation. In quantum homomorphic encryption, *quantum* input data is encrypted in such a way that a server can carry out arbitrary *quantum* computations on the encrypted data, without interacting with the encrypting party. This contrasts *blind* or *delegated* quantum computation where some interaction between client and server is usually required [Chi05; BFK09; ABEM17; VFPR14; FBS+14; Bro15; Lia15; CGJV19].

Yu, Pérez-Delgado and Fitzsimons [YPF14] showed that perfectly information-theoretically secure QFHE is not possible unless the size of the encryption grows exponentially in the input size. Thus, any scheme that attempts to achieve information-theoretically secure QFHE has to leak some proportion of the input to the server [AS06; RFG12] or can only be used to evaluate a subset of all unitary transformations on the input [RFG12; Lia13; TKO+16]. Like the multiplication operation is hard in the classical case, the hurdle in the quantum case seems to be the evaluation of non-Clifford gates. Ouyang, Tan and Fitzsimons provide information-theoretic security for circuits with at most a constant number of non-Clifford operations [OTF15].

The impossibility result by Yu et al. has shifted focus toward constructing quantum homomorphic-encryption schemes using computational assumptions, which may allow bypassing the impossibility and working toward a (quantum) fully-homomorphic-encryption scheme. Broadbent and Jeffery [BJ15] describe a quantum homomorphic-encryption scheme called CL, that is computationally secure for circuits with only Clifford gates. The construction is based on a very natural idea: to encrypt a message qubit under the quantum one-time pad, and to encrypt the (classical) keys to the quantum one-time pad under classical FHE, attaching the ciphertexts to the quantum ciphertext, as in Equation (1.1). An evaluator can perform arbitrary Clifford operations on encrypted qubits, simply by applying the actual Clifford circuit to the ciphertext. Since the Pauli operations that constitute the one-time pad commute with the Clifford group in a known way, the evaluator only needs to (homomorphically) update the classical keys, according to those commutation rules. The scheme CL can be based on any classical FHE scheme, so any advances in classical FHE (e.g., more efficient evaluation) directly improves CL. In fact, the classical key updates required for Clifford gates are only additive: an additively homomorphic classical encryption scheme suffices.

We describe CL in more detail in Section 5.3. It can be regarded as analogous to additively-homomorphic-encryption schemes in the classical setting. The challenge, like multiplication in the classical case, is to perform non-Clifford operations such as the T gate. That will be the topic of Chapter 6.

The construction of quantum homomorphic encryption raises an important question: do the numerous classical applications of FHE have suitable quantum analogues? As it turns out, most of the classical applications require an additional property which is simple classically, but nontrivial quantumly. That property is *verification*: the ability of the user to check that the final ciphertext produced by the server is indeed the result of a particular computation, homomorphically applied to the initial user-generated ciphertext. In the classical case, this is a simple matter: the server makes a copy of each intermediate computation step, and provides the user with all these copies. In the quantum case, such a “transcript” or “log” appears to violate no-cloning.

Verification of quantum computations has been a topic of interest in various contexts over the last few years. Protocols have been designed for outsourcing a quantum computation to multiple (entangled, but noncommunicating) servers [RUV13; GKW15; CGJV19; Gri19], and/or using multiple rounds of communication [BFK09; ABEM17]. Recently, proving an instance to be in BQP was shown to be possible with a single prover and a classical verifier [Mah18b], even noninteractively and in zero-knowledge [ACGH19]. These protocols do not immediately yield a verifiable QFHE scheme, however, since the reduction from a computation to a BQP instance might require multiple rounds of interaction.

5.1.1 Contributions

In this chapter, we lay the ground work for constructing quantum fully homomorphic encryption (see Chapter 6), both without and with verification. Here, we focus on quantum encryption that is homomorphic for *Clifford* circuits only. We outline the framework set up by Broadbent and Jeffery (Section 5.2), and describe the scheme CL in detail (Section 5.3). In terms of new contributions, this chapter contains the following.

Circuit privacy (Section 5.3.1). In homomorphic encryption, the server is allowed to choose which circuit she wants to apply to the (encrypted) data. In some settings, the server may need to keep the evaluated circuit secret from the client. In CL, circuit privacy in the passive setting almost comes for free: the evaluating party can add an extra randomization layer to the output state by applying her own one-time pad. We show that if the classical FHE scheme has circuit privacy, then this extra randomization step provides circuit privacy for CL.

Definition of verifiable QHE (Section 5.4). We define a new primitive: verifiable quantum homomorphic encryption (QHE). A standard QHE scheme consists of four algorithms: KeyGen, Enc, Eval and Dec. We define verifiable QHE similarly, with two changes: (1) Eval provides an extra classical "computation log" output; (2) decryption is now called VerDec, and accepts a ciphertext, a circuit description c, and a computation log. A crucial parameter is the relative difficulty of performing c and VerDec_k^c. In a nontrivial scheme, the latter must be simpler.

Informally, security requires that, if a server deviates significantly from the honest evaluation, then VerDec will reject. We give two alternative definitions of security, that adapt the notions of computational semantic security SEM and computational indistinguishability security IND to the verifiable setting. Generalizing the relation SEM $\Leftrightarrow$ IND [ABF+16], we show that the two definitions are equivalent:

1. **Semantic security (SEM-VER).** Consider a QPT adversary $\mathcal{A}$ which manipulates a ciphertext and declares a circuit. This adversary defines a channel $\Phi_{\mathcal{A}} := \mathsf{VerDec} \circ \mathcal{A} \circ \mathsf{Enc}$. A simulator $\mathcal{S}$ does not receive or output a ciphertext, but does declare a circuit; it defines a channel $\Phi_{\mathcal{S}}$ which first runs $\mathcal{S}$ and then runs a circuit on the plaintext based on the outputs of $\mathcal{S}$. We say that a verifiable QHE scheme is semantically secure (SEM-VER) if for all adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that the channels $\Phi_{\mathcal{A}}$ and $\Phi_{\mathcal{S}}$ are computationally indistinguishable. Restricting SEM-VER to the empty-circuit case, we recover (the

computational version of) the definition of quantum message authentication (see Definition 3.2.1).

2. **Indistinguishability (IND-VER)**. Consider the following security game. Based on a hidden coin flip $r \in \{0,1\}$, $\mathcal{A}$ participates in one of two protocols. For $r = 0$, it is normal verifiable QHE. For $r = 1$, it is a modified execution, where we secretly swap out the plaintext $\rho_{\mathcal{A}}$ to a private register (replacing it with an encryption of a fixed state), apply the desired circuit to $\rho_{\mathcal{A}}$, and then swap $\rho_{\mathcal{A}}$ back in. We then discard this plaintext if VerDec rejects the outputs of $\mathcal{A}$. Upon receiving the final plaintext of the protocol, $\mathcal{A}$ must guess the bit r. A verifiable QHE scheme is IND-VER if no adversary $\mathcal{A}$ has a more than negligible advantage in guessing r.

A verifiable QHE scheme for Clifford gates (Section 5.6). We construct a new QHE scheme where the server can certify, using a classical computation log as "proof", that a particular homomorphic computation was performed on a ciphertext. The verification of the proof is mostly classical; if the output of the quantum circuit is classical, then the verification is entirely so. Unlike all previously known quantum homomorphic-encryption schemes, the underlying encryption is now authenticated.

We briefly sketch the scheme, which is called TCL (for "trap-code CL"). It is based on the ideas of quantum computing on authenticated data for the trap code (see Section 3.4.2). The inputs to the computation are encrypted using the trap code, and the keys to the trap code (the permutation and one-time-pad keys) are encrypted under a classical homomorphic-encryption scheme. Similarly to CL, the evaluator applies gates to the trap code while homomorphically updating the classical keys, and keeps a "log" (i.e., transcript) of all those classical evaluation steps. For some gates (P and H), the evaluator needs encoded magic states to successfully apply the gate: these are supplied by the key generator. At decryption time, the classical FHE transcript is checked for consistency, and the resulting keys are used to decrypt the trap-code-encoded output state, outputting either the plaintext or a reject flag.

Our scheme TCL is compact: the number of elementary quantum operations performed by the verified-decryption function scales only with the size of the plaintext, and not with the size of the circuit. We do require a classical computation which can scale with the size of the circuit; this is reasonable since the decryption function must receive a description of the circuit as an input.

As a building block for TCL, we first construct a noncompact scheme TC (Section 5.5), which is homomorphic only for a small number of gates, but does not rely on classical homomorphic encryption. As such, TC is information-theoretically secure.

For an overview of the schemes discussed in this chapter and the next, refer to Figure A on page 289.

5.2 Homomorphic encryption

We start by defining (classical and quantum) homomorphic encryption more formally, and specifying the security conditions for such schemes. We will focus on the public-key setting, and briefly comment on how to adapt the definitions for the symmetric-key setting. Broadbent and Jeffery [BJ15] treat the variations on the standard public-key definition more thoroughly.

5.2.1 The classical setting

A classical homomorphic-encryption scheme HE consists of four algorithms: key generation, encryption, evaluation, and decryption. The key generator produces three keys: a public key and evaluation key, both of which are publicly available to everyone, and a secret key which is only revealed to the decrypting party. Anyone in possession of the public key can encrypt the inputs $x_1, \ldots, x_\ell$, and send the resulting ciphertexts $y_1, \ldots, y_\ell$ to an evaluator who evaluates some circuit c on them. The evaluator sends the result to a party that possesses the secret key, who should be able to decrypt it to $c(x_1, \ldots, x_\ell)$.

Traditionally, the evaluation key is considered as part of the public key, since it is itself publicly known [Gen09]. We purposefully distinguish between the two, because in the quantum setting we will allow only the evaluation key to be a quantum state. It becomes a resource that is consumed during the evaluation, while the public key can in principle be reused to securely encrypt other messages.

Definition 5.2.1 (Classical homomorphic encryption [BV14])**.** A classical homomorphic-encryption scheme HE consists of four algorithms, which run in classical probabilistic polynomial time in terms of their input and parameters:

Key generation: $(pk, evk, sk) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$, where $\lambda \in \mathbb{N}$ is the security parameter. Three keys are generated: a public key pk, which can be used for the encryption of messages; a secret key sk used for decryption; and an evaluation key evk that may aid in evaluating the circuit on the encrypted state. The keys pk and evk are announced publicly, while sk is kept secret.

Encryption: $y \leftarrow \mathsf{HE.Enc}_{pk}(x)$ for some one-bit message $x \in \{0,1\}$. This probabilistic procedure outputs a ciphertext y, using the public key pk.

Evaluation: $y' \leftarrow \mathsf{HE.Eval}^{c}_{evk}(y_1, \ldots, y_\ell)$ uses the evaluation key to output some ciphertext y' which decrypts to the evaluation of circuit c on the plaintexts for $y_1, \ldots, y_\ell$. We will often think of Eval as an evaluation of a function f instead of some canonical circuit for f, and write $\mathsf{HE.Eval}^{f}_{evk}(y_1, \ldots, y_\ell)$ in this case.

Decryption: $x' \leftarrow \mathsf{HE.Dec}_{sk}(y')$ outputs a message $x' \in \{0,1\}^*$, using the secret key sk.

The way it is defined, $\mathsf{HE.Enc}_{pk}$ can only encrypt single bits. When encrypting an n-bit message $x \in \{0,1\}^n$, we encrypt the message bit-by-bit, applying the encryption procedure n times. We abuse the notation $\mathsf{HE.Enc}_{pk}(x)$ to denote this bitwise encryption of the string x. Note that Definition 5.2.1 does not require the decryption to work bit-by-bit: if we do require every bit of x' to be recoverable from separate parts of y', the homomorphic-encryption scheme is called *divisible*.

Of course, we want the output plaintext x' to be equal to $c(x_1, \ldots, x_\ell)$, the application of c to the input plaintexts. This property is captured by the following definition.

Definition 5.2.2 (Correctness)**.** A classical $\mathcal{C}$-homomorphic-encryption scheme HE is correct (for a circuit class $\mathcal{C}$) if for every circuit $c \in \mathcal{C}$, input x, and key set $(pk, evk, sk) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$,

$$\Pr[\mathsf{HE.Dec}_{sk}(\mathsf{HE.Eval}^c_{evk}(\mathsf{HE.Enc}_{pk}(x))) \neq c(x)] \leq \mathrm{negl}(\lambda)\,. \tag{5.1}$$

For clarity of exposition, we assume that the classical schemes HE we use are perfectly correct, and that it is possible to immediately decrypt after encrypting (implicitly evaluating the identity circuit). We will use the notation $\widetilde{x}$ to denote any ciphertext for x: that can be the result of running $\mathsf{HE.Enc}_{pk}(x)$, but also the output of an evaluation. In any case, $\widetilde{x}$ is such that $\mathsf{Dec}_{sk}(\widetilde{x}) = x$ with overwhelming probability.

A homomorphic-encryption scheme HE is secure if the underlying encryption scheme is secure under chosen-plaintext attacks by quantum adversaries:

Definition 5.2.3 (q-IND-CPA security of classical HE [BJ15, Definition 3.1])**.** A classical homomorphic-encryption scheme HE is q-IND-CPA secure (quantum indistinguishability under chosen-plaintext attacks) if for any quantum polynomial-time adversary $\mathcal{A}$ and for $(pk, evk, sk) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$,

$$\left|\Pr\left[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(0)) = 1\right] - \Pr\left[\mathcal{A}(pk, evk, \mathsf{HE.Enc}_{pk}(1)) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

Given any nonhomomorphic-encryption scheme $\mathsf{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, we can trivially build a homomorphic-encryption scheme, as follows. Define the scheme TRIV by setting key generation and encryption to be identical to the nonhomomorphic-encryption scheme (the key generation does not generate an evaluation key; it is left empty). Then, define $\mathsf{TRIV.Eval}^c_{evk}(y_1, \ldots, y_\ell) := (y_1, \ldots, y_\ell, c)$ to be the function that simply appends a description of the circuit c to the ciphertexts, without performing any computation on them. Finally, define

$$\mathsf{TRIV.Dec}_{sk}(y_1, \ldots, y_\ell, c) := c\big(\mathsf{E.Dec}(y_1), \ldots, \mathsf{E.Dec}(y_\ell)\big)\,, \tag{5.2}$$

that is, the decryption function decrypts the input ciphertexts, and computes the circuit c itself.

Although TRIV satisfies Definitions 5.2.1 to 5.2.3, its spirit is clearly not what one wants in homomorphic encryption: namely, to outsource a computation to an evaluator. In TRIV, the entire computational burden of the circuit c lies with the decrypting party. To enforce transferring the computational burden from the client to the server, we aim for a homomorphic-encryption schemes to be *compact*, requiring that the complexity of the decryption function does not depend on the size of the circuit:

Definition 5.2.4 (Compactness)**.** A classical homomorphic-encryption scheme HE is compact if there exists a polynomial $p(\lambda)$ such that for any circuit c with n_{out} output bits, and any input x, the complexity of applying $\mathsf{HE.Dec}$ to the result of $\mathsf{HE.Eval}^c_{evk}(\mathsf{HE.Enc}_{pk}(x))$ is at most $p(\lambda, n_{\mathsf{out}})$.

A scheme that is both correct for all circuits and compact, is called *fully* homomorphic (FHE). If it is only correct for a subset of all possible circuits (e.g., all circuits with no multiplication gates) or if it is not compact, it is considered to be a *somewhat* homomorphic or *partially* homomorphic scheme.

Often, we wish to consider FHE schemes which require an a priori upper bound (polynomial in the security parameter) on the depth of circuits to be homomorphically evaluated [Vai11]. In the current state of the art, such schemes (referred to as *leveled* FHE) can be constructed under milder assumptions than unleveled schemes: in particular, they do not require circular-security-type assumptions. There are a few variants of leveled FHE defined in the literature. We use the following.

Definition 5.2.5 (Leveled FHE)**.** A leveled FHE scheme is a scheme where the key generation takes an additional parameter $\mathsf{KeyGen}(1^\lambda, 1^d)$ and outputs (pk, evk, sk). Correctness holds only for evaluating circuits of total depth at most d. Furthermore, the length of sk and the complexity of decryption are independent of d.

Secure homomorphic encryption, in the sense of Definition 5.2.3, ensures the privacy of the input data. It does not guarantee, however, that whoever generates the keys, encrypts, and decrypts cannot gain information about the circuit c that was applied to the input by the evaluator. Obviously, the output value $c(x)$ often reveals something about the circuit c, but apart from this necessary leakage of information, one may require a homomorphic-encryption scheme to ensure *circuit privacy* in the sense that an adversary cannot statistically gain any information about c from the output of the evaluation procedure that it could not already gain from $c(x)$ itself. In the definition below, we consider passive adversaries, that follow the protocol honestly but try to learn from the information they receive in the protocol.

Definition 5.2.6 (Statistical circuit privacy in the semi-honest setting [IP07])**.** A classical homomorphic-encryption scheme HE has statistical circuit privacy in the semi-honest ("honest-but-curious") model if there exists a PPT algorithm $\mathcal{S}_{\mathsf{HE}}$ such that for any security parameter λ, input x, keys $(pk, evk, sk) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$, and circuit c:

$$\mathsf{HE.Eval}^c_{evk}\big(\mathsf{HE.Enc}_{pk}(x)\big) \approx_{\mathrm{negl}(\lambda)} \mathcal{S}_{\mathsf{HE}}(1^\lambda, pk, evk, c(x)).$$

That is, the statistical distance (see Section 2.2.3) is upper bounded by $\mathrm{negl}(\lambda)$.

5.2.2 The quantum setting

A quantum homomorphic-encryption scheme QHE is a natural extension of the classical case, and differs from it in only a few aspects [BJ15]. The secret and public keys are still classical, but the evaluation key is now allowed to be a quantum state. This means that the evaluation key is not necessarily reusable, and can be consumed during the evaluation procedure. The messages to be encrypted can be qubits instead of bits, and the evaluator should be able to evaluate quantum circuits on them.

Definition 5.2.7 (Quantum homomorphic encryption [BJ15])**.** A quantum homomorphic-encryption scheme QHE consists of four algorithms, which run in quantum polynomial time in terms of their input and parameters:

Key generation: $(pk, \rho_{evk}, sk) \leftarrow \mathsf{QHE.KeyGen}(1^\lambda)$, where $\lambda \in \mathbb{N}$ is the security parameter. In contrast to the classical case, the evaluation key is a quantum state.

Encryption: $\sigma \leftarrow \mathsf{QHE.Enc}_{pk}(\rho)$ produces, for every valid public key pk and input state ρ from some message space, a quantum ciphertext σ in some cipherspace.

Evaluation: $\sigma' \leftarrow \mathsf{QHE.Eval}^{c}_{\rho_{evk}}(\sigma)$ represents the evaluation of a circuit c. If c requires n input qubits, then σ should be a product of n ciphertexts. The evaluation function maps it to a product of n' states in some output space, where n' is the number of qubits that c would output. The evaluation key ρ_{evk} is consumed in the process.

Decryption: $\rho' \leftarrow \mathsf{QHE.Dec}_{sk}(\sigma')$ maps a single ciphertext σ' from the output space to a single-qubit quantum state ρ' in the message space. Note that if the evaluation procedure QHE.Eval outputs a product of n' states, then QHE.Dec needs to be run n' times.

The decryption procedure differs from the classical definition in that we require the decryption to happen subsystem-by-subsystem: this is fundamentally different from the more relaxed notion of *indivisible schemes* [BJ15] where an auxiliary quantum register may be built up for the entire state, and the state can only be decrypted as a whole. In this work, we only consider the divisible definition.

Correctness in the quantum setting is as in Definition 5.2.2, except that we require the trace distance between the decrypted state and the ideal state $\Phi_c(\rho)$ (rather than the probability that they differ) to be negligible. The notions of compactness, partial/fully homomorphic encryption, and leveled homomorphic encryption carry over unchanged.

In terms of security, we again aim for indistinguishability under chosen-plaintext attacks, where the attacker may have quantum computational powers (q-IND-CPA). The difference with the classical setting described in Section 2.3.2 is that the plaintext message may be quantum, and therefore may be entangled with an environment.

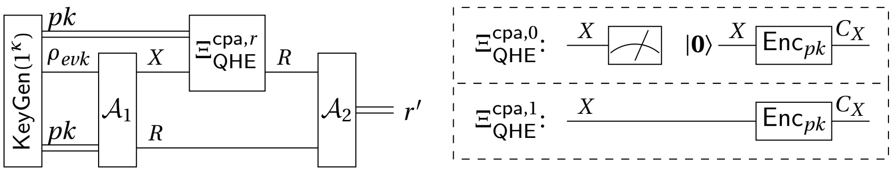


Figure 5.1: [BJ15, reproduced with permission of the authors] The quantum CPA indistinguishability experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{QHE}}(\lambda)$. Double lines represent classical information flow, and single lines represent quantum information flow. The adversary $\mathcal{A}$ is split up into two separate algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, which share a working memory represented by the quantum state in the register R.

Security is captured by a *security game* which, if no adversary can have a significant advantage to win the game, ensures semantic security [ABF+16]. We restate it here for completeness.

Game 5.2.8 (quantum CPA indistinguishability game [BJ15])**.** The q-IND-CPA game with respect to a scheme QHE and a quantum polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, denoted by $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{QHE}}(\lambda)$, is defined by the following procedure:

1. $\mathsf{KeyGen}(1^\lambda)$ is run to obtain keys (pk, sk, ρ_{evk}).
2. Adversary $\mathcal{A}_1$ is given (pk, ρ_{evk}) and outputs a quantum state on the registers XR (these are the message register, and a reference register containing any side information).
3. For $r \in \{0,1\}$, let $\Xi^{\mathsf{cpa},r}_{\mathsf{QHE}}$ be defined as:
$$\Xi^{\mathsf{cpa},0}_{\mathsf{QHE}}(pk,\rho) := \mathsf{QHE.Enc}_{pk}(|0\rangle\langle 0|), \tag{5.3}$$
$$\Xi^{\mathsf{cpa},1}_{\mathsf{QHE}}(pk,\rho) := \mathsf{QHE.Enc}_{pk}(\rho). \tag{5.4}$$
A random bit $r \in \{0,1\}$ is chosen and $\Xi^{\mathsf{cpa},r}_{\mathsf{QHE}}$ is applied to the state in X (the output being a state in C_X).
4. Adversary $\mathcal{A}_2$ obtains the system in $C_X R$ and outputs a bit r'.
5. The output of the experiment is defined to be 1 if $r' = r$ and 0 otherwise. In the output is 1, we say that $\mathcal{A}$ *wins* the experiment.

The game $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{QHE}}(\lambda)$ is depicted in Figure 5.1. Informally, the challenger randomly chooses whether to encrypt some message, chosen by the adversary, or instead to encrypt the state $|0\rangle\langle 0|$. The adversary has to guess which of the two happened. If he cannot do so with more than negligible advantage, the encryption procedure is considered to be q-IND-CPA secure:

Definition 5.2.9 (q-IND-CPA security of quantum HE [BJ15, Definition 3.3])**.** A (classical or quantum) homomorphic-encryption scheme S is *q-IND-CPA* secure if for any quantum polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{S}}(\lambda) = 1] \leqslant \frac{1}{2} + \text{negl}(\lambda).$$

Analogously to $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{S}}(\lambda)$, we can define a game $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A},\mathsf{S}}(\lambda)$, in which the adversary can give multiple messages to the challenger. The messages are either all encrypted, or all replaced by zeros. Broadbent and Jeffery [BJ15] show that these notions of security are equivalent.

For a private-key scheme, q-IND-CPA security is defined analogously, but based on the indistinguishability experiment $\mathsf{SymK}_{\mathcal{A},\mathsf{S}}(\lambda)$ [BJ15]: this game is the same as Game 5.2.8, except that the adversary $\mathcal{A}_1$ is not given the encryption key *sk* (*pk* in the public-key game). Instead, $\mathcal{A}_1$ gets access to an *encryption oracle*, to which it can submit any number of plaintexts, and receive the corresponding ciphertexts, before having to select a challenge plaintext.

To define circuit privacy in the quantum setting, we need to take into account the fact that the input state may be part of some larger (possibly entangled) system. This leads to the following definition of *quantum circuit privacy* in the semi-honest setting:

Definition 5.2.10 (Quantum circuit privacy in the semi-honest setting)**.** A quantum homomorphic-encryption scheme QHE has statistical circuit privacy in the semi-honest setting if there exists a QPT algorithm $\mathcal{S}_{\mathsf{QHE}}$ such that for any security parameter λ, keys $(pk, \rho_{evk}, sk) \leftarrow \mathsf{QHE.KeyGen}(1^\lambda)$, and circuit c:

$$\left\| \mathsf{QHE.Eval}^{c}_{\rho_{evk}}\left(\mathsf{QHE.Enc}_{pk}(\cdot)\right) - \mathcal{S}_{\mathsf{QHE}}\left(1^\lambda, pk, evk, \Phi_c(\cdot)\right) \right\|_\diamond \leqslant \text{negl}(\lambda).$$

There are various ways to define passive adversaries in a quantum setting [DNS10; BB14]. In Definition 5.2.10, we consider adversaries that follow all protocol instructions exactly.

5.3 CL: a partially-homomorphic scheme

All quantum homomorphic-encryption schemes discussed in this work are extensions of a basic scheme CL [BJ15], an encryption that is homomorphic for all Clifford circuits. The CL scheme can be regarded as analogous to additively-homomorphic-encryption schemes in the classical setting. In this section we discuss it and its properties as a warm-up for the definition and analysis of more complicated protocols later on.

CL relies on two ingredients: the information-theoretically secure quantum one-time pad, and a post-quantum secure classical FHE scheme (which necessarily relies on computational assumptions). Input quantum states are encrypted using the quantum one-time pad, and the Pauli keys are themselves encrypted under the classical FHE. CL allows an evaluator to compute arbitrary Clifford operations on encrypted qubits, simply by performing the actual Clifford circuit, followed by homomorphically updating the quantum one-time pad keys according to the commutation rules between the performed Clifford gates and the Pauli encryptions.

In Protocol 5.3.1, we assume that the evaluated circuit c is given in terms of the Clifford generator set $\{\mathsf{P}, \mathsf{H}, \mathsf{CNOT}\}$. This allows us to evaluate c gate-by-gate, each time only affecting one or two wires. We specify the evaluation procedure only for the gates P, H, and CNOT: a larger circuit is evaluated by concatenating the evaluation procedures for those basic gates[1].

Protocol 5.3.1 (CL: quantum homomorphic encryption for Clifford circuits)**.** Let HE be a classical fully-homomorphic-encryption scheme. The quantum homomorphic-encryption scheme CL is defined by the following algorithms:

Key generation ($\mathsf{CL.KeyGen}(1^\lambda)$)**.** Run $(pk, evk, sk) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$, and output $(pk, |evk\rangle, sk)$.

Encryption ($\mathsf{CL.Enc}_{pk}(\rho)$)**.** A single-qubit state ρ is encrypted with a quantum one-time pad, and the pad key is classically encrypted and appended to the state, resulting in the classical-quantum state

$$\frac{1}{4}\sum_{a,b\in\{0,1\}} \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a \otimes \big|\mathsf{HE.Enc}_{pk}(a,b)\big\rangle\big\langle\mathsf{HE.Enc}_{pk}(a,b)\big|. \tag{5.5}$$

We will call the qubits on the left-hand side of the tensor product the data qubits, and those on the right-hand side the (encrypted) key state. Recall that we abuse the notation $\mathsf{HE.Enc}_{pk}(a,b)$ to denote the separate encryptions of a and b. Anyone holding the state in Equation (5.5) has access to the individual (classical) ciphertexts $\widetilde{a}$ and $\widetilde{b}$.

Phase-gate evaluation ($\mathsf{CL.Eval}^{\mathsf{P}}_{|evk\rangle}(\sigma)$)**.** Apply P to the relevant data qubit of σ, and apply the following key update to the key state:

$$\mathsf{P}\text{-update} : (a,b) \mapsto (a, b\oplus a). \tag{5.6}$$

[1]Pauli gates can be implemented using P and H, since $\mathsf{Z} = \mathsf{P}^2$ and $\mathsf{X} = \mathsf{HP}^2\mathsf{H}$. However, it is more efficient to implement them directly: the evaluation of an X gate, for example, leaves the quantum state untouched, and classically updates the key a to $a \oplus 1$ using HE.Eval.

Since σ only contains encryptions $\widetilde{a}$ and $\widetilde{b}$, the key update is performed by evaluating $\mathsf{HE.Eval}_{evk}^{\text{P-update}}(\widetilde{a},\widetilde{b})$.

Hadamard evaluation ($\mathsf{CL.Eval}_{|evk\rangle}^{\mathsf{H}}(\sigma)$)**.** Similar to phase-gate evaluation, except with the key-update function

$$\mathsf{H}\text{-update}: (a,b) \mapsto (b,a). \tag{5.7}$$

Controlled-not evaluation ($\mathsf{CL.Eval}_{|evk\rangle}^{\mathsf{CNOT}}(\sigma)$)**.** Since CNOT is a two-qubit gate, apply it jointly to the *two* relevant data qubits of σ, which each have their own independent one-time pad keys (a_1, b_1) and (a_2, b_2), encryptions of which are stored in the key state. Homomorphically apply the classical key-update function

$$\mathsf{CNOT}\text{-update}: (a_1, b_1, a_2, b_2) \mapsto (a_1, b_1 \oplus b_2, a_1 \oplus a_2, b_2). \tag{5.8}$$

Measurement evaluation ($\mathsf{CL.Eval}_{|evk\rangle}^{\nearrow}(\sigma)$)**.** For a computational-basis measurement, measure the encrypted qubit σ in the computational basis, resulting in a single measurement bit m.

Decryption ($\mathsf{CL.Dec}_{sk}(\sigma)$)**.** For each data qubit, run $\mathsf{HE.Dec}_{sk}$ twice in order to retrieve the updated keys a' and b' to the quantum one-time pad. Apply the Pauli decryption $\mathsf{X}^{a'}\mathsf{Z}^{b'}$ to the data qubit. For measured qubits, compute $m \oplus a'$ to retrieve the plaintext measurement result.

To see why the gate evaluations are correct, we work out the case for the phase gate in detail. The two other cases work similarly. When the phase-gate evaluation procedure is applied to an input state of the form

$$\sigma = \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a \otimes \left|\widetilde{a},\widetilde{b}\right\rangle\!\left\langle\widetilde{a},\widetilde{b}\right|, \tag{5.9}$$

the resulting state is

$$\mathsf{P}\mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a\mathsf{P}^\dagger \otimes \left|\mathsf{HE.Eval}_{evk}^{\text{P-update}}(\widetilde{a},\widetilde{b})\right\rangle\!\left\langle\mathsf{HE.Eval}_{evk}^{\text{P-update}}(\widetilde{a},\widetilde{b})\right| \tag{5.10}$$

$$= \mathsf{X}^a\mathsf{Z}^{b\oplus a}\mathsf{P}\rho\mathsf{P}^\dagger\mathsf{Z}^{b\oplus a}\mathsf{X}^a \otimes \left|\widetilde{a},\widetilde{b\oplus a}\right\rangle\!\left\langle\widetilde{a},\widetilde{b\oplus a}\right|. \tag{5.11}$$

The equality follows from (perfect) correctness of the classical homomorphic-encryption scheme, and the commutation relation between P and the Pauli group (see Section 2.4.2).

CL is correct (for all Clifford+measurement circuits), compact, and secure, as long as the underlying classical homomorphic encryption has those properties [BJ15]. To provide intuition for the upcoming protocols, we briefly sketch security here. A similar proof is given for Lemmas 5.6.11 and 6.3.4.

The security of CL is proven in two steps: the first relies on the security of HE, the second on that of the quantum one-time pad.

In the first step, it is shown that no adversary $\mathcal{A}$ can have a significantly better winning probability in the q-IND-CPA game (Game 5.2.8) when given an encrypted state as in Equation (5.5), as opposed to an encryption of the form

$$\frac{1}{4}\sum_{a,b\in\{0,1\}} \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a \otimes \left|\mathsf{HE.Enc}_{pk}(0,0)\right\rangle\!\left\langle\mathsf{HE.Enc}_{pk}(0,0)\right|, \tag{5.12}$$

where the key state is replaced with encryptions of zeros. If $\mathcal{A}$ *does* have a significant advantage in the first game, then it can be used as a subroutine to win the q-IND-CPA game against the classical scheme HE. Essentially, an adversary $\mathcal{A}'$ for the HE game plays the role of the challenger against $\mathcal{A}$. In the encryption step, after sampling keys (a,b), the "challenger" $\mathcal{A}'$ submits the keys to its own challenger, so that the encryption it returns to $\mathcal{A}$ is either of the form of Equation (5.5) or Equation (5.12). If $\mathcal{A}$ answers the challenge posed by $\mathcal{A}'$ correctly, then $\mathcal{A}'$ will assume that the keys (a,b) really were encrypted. Otherwise, $\mathcal{A}'$ will assume that its own challenger encrypted zeros. If $\mathcal{A}$ indeed has a significantly better winning probability whenever it receives an encryption like Equation (5.5), then this strategy gives $\mathcal{A}'$ a significant advantage in guessing the actions of its own challenger.

The second step is to observe that the quantum part of an encryption of the form of Equation (5.12) is a quantum one-time pad. Hence, no adversary $\mathcal{A}$ can have an advantage in Game 5.2.8 when given encryptions of the form of Equation (5.12).

5.3.1 Circuit privacy

By itself, CL does not provide circuit privacy in the sense of Definition 5.2.10, even if the underlying classical scheme HE is circuit private. As a counterexample, consider the input $|0\rangle\langle 0|$, encrypted with Pauli keys (a,b). On the plaintext level, there is no difference between the identity circuit, and the single-gate circuit P: in both cases, the plaintext output is $|0\rangle\langle 0|$. However, the decrypting party can tell the difference between the two by looking at the decrypted Pauli keys: if identity was applied, the keys will still be (a,b), whereas if P was applied, they will be $(a,b\oplus a)$. Even though the difference in keys does not matter for the plaintext value of the output, they reveal information about the circuit that cannot be deduced from the decrypted output by itself.

Nonetheless, CL can straightforwardly be adapted to a quantum homomorphic-encryption scheme CL$'$ *with* circuit privacy. At the end of the evaluation, the evaluator simply has to apply a freshly random quantum one-time pad to the (already encrypted) evaluation result, and update the classical keys accordingly. The keys themselves are uniformly random, and therefore do not reveal any information about the circuit. Circuit privacy of HE ensures that even the classical encryption of the keys contains no information about the circuit.

Theorem 5.3.2. *If* HE *has circuit privacy in the semi-honest setting, then* CL' *is a quantum homomorphic-encryption scheme with circuit privacy.*

Proof. For notational convenience, we will write $\mathcal{S}(c(x))$ instead of $\mathcal{S}(1^\lambda, pk, evk, c(x))$ in case all other variables are clear. Furthermore, for simplicity we will assume that the computation does not contain any measurements. The same proof, but with slightly lengthier notation, goes through if measurements are part of the computation.

Let $\mathcal{S}_{\mathsf{HE}}$ be the classical simulator guaranteed to exist by the classical circuit privacy of HE (see Definition 5.2.6). Define the simulator $\mathcal{S}_{\mathsf{CL}'}$ (for an n-qubit quantum state σ in a message register M) as

$$\mathcal{S}_{\mathsf{CL}'}(1^\lambda, pk, |evk\rangle, \sigma^M) := \frac{1}{2^{2n}} \sum_{x,z\in\{0,1\}^n} \mathsf{X}^x\mathsf{Z}^z\sigma\mathsf{Z}^z\mathsf{X}^x \otimes |\mathcal{S}_{\mathsf{HE}}(x,z)\rangle\langle\mathcal{S}_{\mathsf{HE}}(x,z)|. \tag{5.13}$$

We claim that $\mathcal{S}^M_{\mathsf{CL}'}(c^M(\rho^{MR}))$ is negligibly close to $\mathsf{CL}'.\mathsf{Eval}^{c\,M}_{|evk\rangle}\left(\mathsf{CL}'.\mathsf{Enc}^M_{pk}(\rho^{MR})\right)$ for any ρ^{MR}, where R is a side-information register with which the input message may be entangled. During $\mathsf{CL}'.\mathsf{Eval}$, the evaluator updates the keys to the quantum one-time pad for all n qubits in the circuit, homomorphically evaluating the appropriate key-update function f_i for each gate i. Write $(d_{a,b}, e_{a,b}) := (f_{|c|}\circ\cdots\circ f_2\circ f_1)(a,b)$, where $|c|$ is the number of gates in c, and abbreviate $g_{c,x} : (a,b) \mapsto d_{a,b}\oplus x$ (and $h_{c,z}$ accordingly for the Z key). Then the state $\mathsf{CL}'.\mathsf{Eval}^c_{|evk\rangle}\left(\mathsf{CL}'.\mathsf{Enc}_{pk}(\rho)\right)$ equals

$$\frac{1}{2^{4n}} \sum_{a,b,x,z\in\{0,1\}^n} \left(\mathsf{X}^{d_{a,b}\oplus x}\mathsf{Z}^{e_{a,b}\oplus z}\right)^M \Phi^M_c(\rho^{MR})\mathsf{Z}^{e_{a,b}\oplus z}\mathsf{X}^{d_{a,b}\oplus x}\otimes$$
$$\left|\mathsf{HE}.\mathsf{Eval}^{(g_{c,x},h_{c,z})}_{evk}\left(\mathsf{HE}.\mathsf{Enc}_{pk}(a,b)\right)\right\rangle\!\left\langle\mathsf{HE}.\mathsf{Eval}^{(g_{c,x},h_{c,z})}_{evk}\left(\mathsf{HE}.\mathsf{Enc}_{pk}(a,b)\right)\right| \tag{5.14}$$
$$\approx \frac{1}{2^{4n}} \sum_{a,b,x,z\in\{0,1\}^n} \left(\mathsf{X}^{d_{a,b}\oplus x}\mathsf{Z}^{e_{a,b}\oplus z}\right)^M \Phi^M_c(\rho^{MR})\mathsf{Z}^{e_{a,b}\oplus z}\mathsf{X}^{d_{a,b}\oplus x}\otimes$$
$$\left|\mathcal{S}_{\mathsf{HE}}\left(d_{a,b}\oplus x, e_{a,b}\oplus z\right)\right\rangle\!\left\langle\mathcal{S}_{\mathsf{HE}}\left(d_{a,b}\oplus x, e_{a,b}\oplus z\right)\right| \tag{5.15}$$
$$= \frac{1}{2^{2n}} \sum_{x,z\in\{0,1\}^n} \left(\mathsf{X}^x\mathsf{Z}^z\right)^M \Phi^M_c(\rho^{MR})\mathsf{Z}^z\mathsf{X}^x \otimes |\mathcal{S}_{\mathsf{HE}}(x,z)\rangle\langle\mathcal{S}_{\mathsf{HE}}(x,z)|. \tag{5.16}$$

The approximation is in terms of trace norm, holds up to a factor of $\mathrm{negl}(\lambda)$, and follows from classical circuit privacy of HE. The final equality is due to the fact that x and z randomize the quantum one-time pad keys, so the keys $(d_{a,b}\oplus x, e_{a,b}\oplus z)$ are themselves uniformly random.

Note that Equation (5.16) equals $\mathcal{S}^M_{\mathsf{CL}'}(\Phi^M_c(\rho^{MR}))$, completing the proof. □

5.4 A new primitive: verifiable quantum homomorphic encryption

In this section, we propose a new definition of verifiable quantum homomorphic encryption (or vQHE), in the symmetric-key setting. We start by expanding the basic definition of QHE, replacing the decryption procedure Dec by a "verify-and-decrypt" procedure VerDec, which potentially rejects the output ciphertext, if it is not the result of the desired computation. Then, we discuss the definitions of compactness and security in this setting. We propose two alternative definitions of security, which we prove to be equivalent.

The definition of vQHE has two parameters: the class $\mathcal{C}$ of circuits which the user can verify, and the class $\mathcal{V}$ of circuits which the user needs to perform in order to verify. We are interested in cases where $\mathcal{C}$ is stronger than $\mathcal{V}$.

Definition 5.4.1 (Verifiable quantum homomorphic encryption)**.** Let $\mathcal{C}$ and $\mathcal{V}$ be (possibly infinite) collections of quantum circuits. A $(\mathcal{C},\mathcal{V})$-vQHE scheme S is a set of four QPT algorithms:

Key generation: $(sk, \rho_{evk}) \leftarrow S.\mathsf{KeyGen}(1^{\lambda})$, where $\lambda \in \mathbb{N}$ is the security parameter. Again, the evaluation key is a quantum state.

Encryption: $\sigma \leftarrow S.\mathsf{Enc}_{sk}(\rho)$ produces a quantum ciphertext in the space $\mathfrak{D}(\mathcal{H}_C)$, given a quantum plaintext in the space $\mathfrak{D}(\mathcal{H}_X)$, and a secret key.

Evaluation: $(log, \sigma') \leftarrow S.\mathsf{Eval}^{c}_{\rho_{evk}}(\sigma)$ represents the evaluation of a circuit $c \in \mathcal{C}$. In addition to the output ciphertext, the evaluation produces a classical computation log log, which will count as a proof that the circuit c was honestly evaluated.

Decryption: $\rho' \otimes |flag\rangle\langle flag| \leftarrow S.\mathsf{VerDec}_{sk}(c, log, \sigma')$ decodes a ciphertext σ' in the space $\mathfrak{D}(\mathcal{H}_C)$ into a plaintext ρ' in the space $\mathfrak{D}(\mathcal{H}_X)$. It additionally appends a *flag* qubit, which is in one of the states $|\mathsf{acc}\rangle\langle\mathsf{acc}|$ or $|\mathsf{rej}\rangle\langle\mathsf{rej}|$. The circuits for $S.\mathsf{VerDec}$ must belong to the class $\mathcal{V}$.

In the above definition, the (classical and quantum) registers are implicitly infinite families of registers, each consisting of poly(λ)-many (qu)bits. In some later definitions, it will be convenient to assume that VerDec also outputs a copy of the (classical) description of the circuit c that it verified.

Similarly to Definition 5.2.2, we want verifiable homomorphic-encryption schemes to yield the correct outcome if executed honestly. In this setting, correctness also requires that an honest execution causes VerDec to accept with high probability.

Definition 5.4.2 (Correctness)**.** A $(\mathcal{C},\mathcal{V})$-verifiable quantum homomorphic-encryption scheme S is correct if for every circuit $c \in \mathcal{C}$, input state ρ^{XR} (where R is some side-information register), and all keys $(sk, \rho_{evk}) \leftarrow S.\mathsf{KeyGen}(1^\lambda)$,

$$\left\| S.\mathsf{VerDec}^{X}_{sk}\left(c, S.\mathsf{Eval}^{c}_{\rho_{evk}}\left(S.\mathsf{Enc}^{X}_{sk}(\rho)\right)\right) - \Phi^{X}_{c}(\rho) \otimes |\mathsf{acc}\rangle\langle\mathsf{acc}| \right\|_{\mathrm{tr}} \leq \mathrm{negl}(\lambda).$$

5.4.1 Compactness

There are trivial vQHE schemes for some choices of $(\mathcal{C}, \mathcal{V})$: for example, if $\mathcal{C} \subseteq \mathcal{V}$, then the user can simply authenticate the ciphertext and then perform the computation during decryption, similarly to the scheme TRIV described in Section 5.2.1. In non-verifiable QHE, such trivial schemes are ruled out by requiring the encryption scheme to be compact (see Definition 5.2.4).

When considering QHE *with* verification, however, some tension arises. On the one hand, trivial schemes like the above still need to be excluded. On the other hand, verifying that a circuit c has been applied requires, at the very least, reading a description of c, which violates the quantum variant of compactness (Definition 5.2.4). Thus, a more careful consideration of the relationship between the desired circuit $c \in \mathcal{C}$ and the verification circuit $V \in \mathcal{V}$ is required. In our work, we will allow the number of *classical* gates in V to scale with the size of c, but only if it is required for verification. We propose a new definition of compactness in this context.

Informally, a vQHE scheme S is compact if $S.\mathsf{VerDec}$ is divisible into a classical verification procedure $S.\mathsf{Ver}$ (outputting only an accept/reject flag), followed by a quantum decryption procedure $S.\mathsf{Dec}$. The running time of $S.\mathsf{Ver}$ is allowed to depend on the circuit size, but the running time of $S.\mathsf{Dec}$ is not. The procedure $S.\mathsf{Dec}$ is not allowed to receive and use any other information from $S.\mathsf{Ver}$ than whether or not it accepts or rejects. This prevents the classical procedure $S.\mathsf{Ver}$ from de facto performing part of the decryption work (e.g., by computing classical decryption keys). In Section 5.5, we will see a scheme that does not fulfill compactness for this reason.

Definition 5.4.3 (Compactness of verifiable QHE)**.** Let S be a verifiable quantum homomorphic-encryption scheme, and write $\sigma' = |y\rangle\langle y| \otimes \sigma''$ for the output quantum state of $S.\mathsf{Eval}$ (that is, σ' contains a, possibly empty, classical component y). The scheme S is compact if the following conditions hold:

1. $S.\mathsf{VerDec}$ can be broken up into subprocedures $S.\mathsf{Ver}$ and $S.\mathsf{Dec}$, as
$$S.\mathsf{VerDec}(c, log, \sigma'', y) = S.\mathsf{Dec}_{sk}\left(\sigma'', y, S.\mathsf{Ver}_{sk}(c, log, y)\right).$$
2. $S.\mathsf{Ver}$ is a classical polynomial-time algorithm that outputs a single flag bit (acc or rej).
3. $S.\mathsf{Dec}$ is a quantum algorithm. There exists a polynomial p such that for any circuit c with n_{out} output qubits, and for any input ρ, $S.\mathsf{Dec}$ runs in time $p(n_{\mathsf{out}}, 1^\lambda)$ on the output of $S.\mathsf{Eval}^{c}_{\rho_{evk}}(S.\mathsf{Enc}_{sk}(\rho))$.

Note that in the above definition, the classical values sk and y are copied and fed to both S.Dec and S.Ver. The classical computation log, the length of which will likely depend on the size of c, is only given to S.Ver. The quantum part of the output ciphertext is only given to the quantum algorithm S.Dec.

5.4.2 Secure verifiability

In this section, we formalize the concept of verifiability. Informally, one would like the scheme to be such that whenever VerDec accepts, the output can be trusted to be close to the desired output. We will consider two *equivalent* formalizations of this idea: a semantic one, and an indistinguishability-based one. We will also show that they imply privacy in the sense of q-IND-CPA (see Definition 5.2.9).

The semantic definition states that every adversary with access to the ciphertext can be simulated by a simulator that only has access to an ideal functionality that simply applies the claimed circuit. It is inspired by quantum authentication [DNS12; BW16] and semantic secrecy [ABF+16].

The real-world scenario (Figure 5.2, top) begins with a state $\rho^{XR_1R_2}$ prepared by a QPT $\mathcal{M}$ (the "message generator") . The register X (plaintext) is subsequently encrypted and sent to the adversary $\mathcal{A}$. The registers R_1 and R_2 contain side information. The adversary acts on the ciphertext and R_1, producing some output ciphertext $C_{X'}$, a circuit description c, and a computation log log. These outputs are then sent to the verified-decryption function. The output, along with R_2, is sent to a distinguisher $\mathcal{D}$, who produces a bit 0 or 1.

In the ideal-world scenario (Figure 5.2, bottom), the plaintext X is not encrypted or sent to the simulator $\mathcal{S}$. The simulator outputs a circuit c and chooses whether to accept or reject. The channel Φ_c implemented by c is applied to the input register X directly. If reject is chosen, the output register X' is traced out and replaced by the fixed state $|\perp\rangle\langle\perp|$; this controlled channel is denoted ctrl-$\oslash$.

Definition 5.4.4 (SEM-VER)**.** A vQHE scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{VerDec})$ is *semantically λ-verifiable* if for any QPT adversary $\mathcal{A}$, there exists a QPT $\mathcal{S}$ such that for all QPTs $\mathcal{M}$ and $\mathcal{D}$,

$$\left|\Pr\left[\mathcal{D}\Big(\mathsf{Real}^{\mathcal{A}}_{sk}(\mathcal{M}(\rho_{evk}))\Big) = 1\right] - \Pr\left[\mathcal{D}\Big(\mathsf{Ideal}^{\mathcal{S}}_{sk}(\mathcal{M}(\rho_{evk}))\Big) = 1\right]\right| \leq \mathrm{negl}(\lambda),$$

where $\mathsf{Real}^{\mathcal{A}}_{sk} = \mathsf{VerDec}_{sk} \circ \mathcal{A} \circ \mathsf{Enc}_{sk}$ and $\mathsf{Ideal}^{\mathcal{S}}_{sk} = \text{ctrl-}\oslash \circ \Phi_c \circ \mathcal{S}_{sk}$, as depicted in Figure 5.2, and the probability is taken over $(\rho_{evk}, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and all QPTs above.

Note that the simulator (in the ideal world) gets the secret key sk. We believe that this is necessary, because the actions of an adversary may depend on superficial properties of the ciphertext. In order to successfully simulate this, the simulator needs to be able to generate (authenticated) ciphertexts. He cannot do so with a fresh

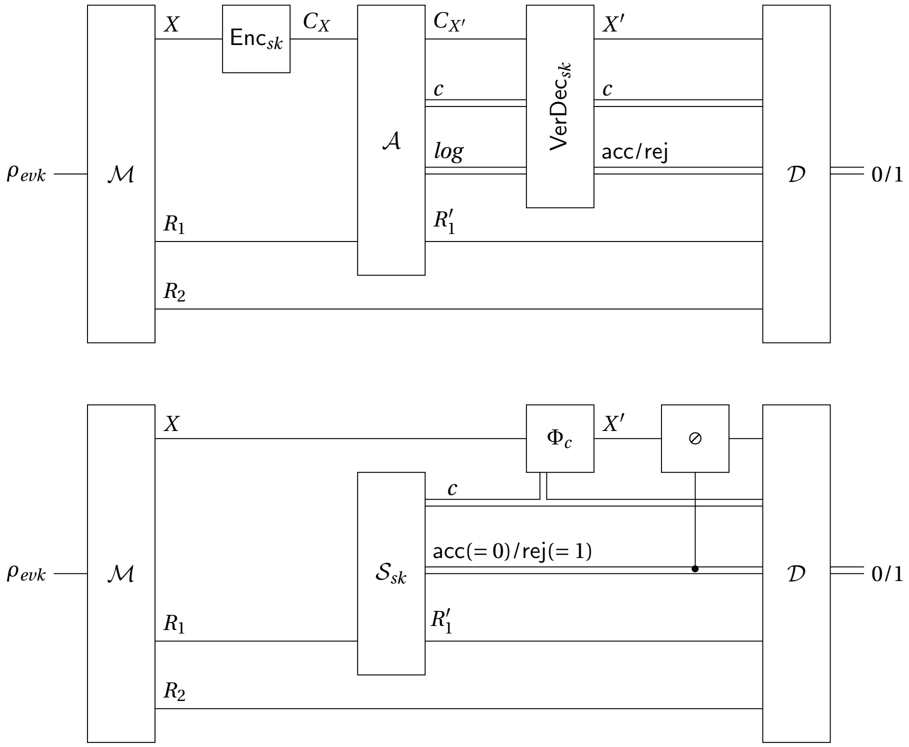


Figure 5.2: The real world (top) and ideal world (bottom) for SEM-VER.

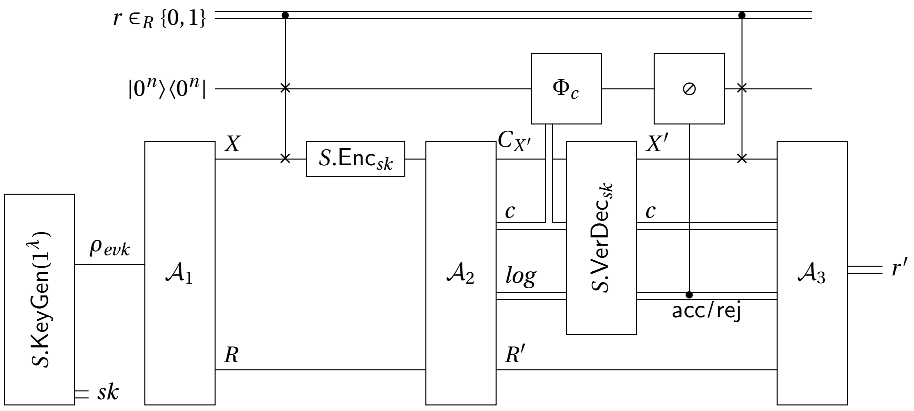


Figure 5.3: The indistinguishability game $\mathsf{VerGame}_{\mathcal{A},S}(\lambda)$, as used in the definition of IND-VER.

secret key, because the input plaintext may depend on the correlated evaluation key ρ_{evk}. Fortunately, the simulator does not become too powerful when in possession of the secret key, because he does not receive any relevant plaintexts or ciphertexts to encrypt or decrypt: the input register X is untouchable for the simulator. Later, in Lemmas 5.4.7 and 5.4.9, we will see that SEM-VER, even with the secret key given to the simulator, implies secure encryption.

Next, we present an alternative definition of verifiability, based on a security game motivated by indistinguishability.

Game 5.4.5. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a scheme S, and a security parameter λ, the $\mathsf{VerGame}_{\mathcal{A},S}(\lambda)$ game proceeds as depicted in Figure 5.3.

The game is played in several rounds. Based on the evaluation key, the adversary first chooses an input (and some side information in R). Based on a random bit r, this input is either encrypted and sent to $\mathcal{A}_2$ (if $r = 0$), or swapped out and replaced by a dummy input $|0^n\rangle\langle 0^n|$ (if $r = 1$). If $r = 1$, the ideal channel Φ_c is applied by the challenger, and the result is swapped back in right before the adversary (in the form of $\mathcal{A}_3$) has to decide on its output bit r'. If $\mathcal{A}_2$ causes a reject, the real result is also erased by the channel $\oslash$. We say that the adversary *wins* (expressed as $\mathsf{VerGame}_{\mathcal{A},S}(\lambda) = 1$) whenever $r' = r$.

Definition 5.4.6 (IND-VER)**.** A vQHE scheme S has *λ-indistinguishable verification* if for any QPT adversary $\mathcal{A}$,

$$\Pr[\mathsf{VerGame}_{\mathcal{A},S}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Definitions 5.4.4 and 5.4.6 both capture the property that an adversary cannot effectively touch the real input state (in the ideal scenario or in the $r = 1$ scenario), and therefore it cannot do so in the real or $r = 0$ scenarios either. This guarantees that the *only* action it can take is the honest one of applying the circuit.

Apart from verifiability, the above definitions also capture *privacy* in the sense of q-IND [ABF+16]: the security game $\mathsf{SymK}_{\mathcal{A},S}(\lambda)$ for q-IND is the same as $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},S}(\lambda)$ (Game 5.2.8 and Figure 5.1), except that the adversary does not receive a public key, because there is none. The idea that verifiability necessarily implies privacy is reminiscent of the analogous result that authentication implies encryption [BCG+02].

To achieve q-IND-CPA security in a symmetric-key setting, the adversary would have to get access to an encryption oracle before the challenge plaintext is chosen [BJ15]. We could straightforwardly adapt Definition 5.4.6 in this manner by giving $\mathcal{A}_1$ access to such an encryption oracle, in which case the definition would imply q-IND-CPA, using a similar proof as below. For simplicity, we will focus on the verifiability definition that provides q-IND security only. Not all protocols presented in this chapter will be q-IND-CPA secure, but those that are can be shown to be so independently of the IND-VER property.

Lemma 5.4.7. *If a vQHE scheme S is IND-VER, then it is q-IND.*

Proof. We will argue that any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that can win the quantum indistinguishability game $\mathsf{SymK}_{\mathcal{A},S}(\lambda)$ (defined above) with nonnegligible advantage can be turned into a successful adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3)$ against the game $\mathsf{VerGame}_{\mathcal{A}',S}(\lambda)$.

The adversary $\mathcal{A}'$ is defined as follows. $\mathcal{A}'_1$ simply runs $\mathcal{A}_1$ on its input, and outputs whatever it outputs into the registers XR. $\mathcal{A}'_2$ runs $\mathcal{A}_2$, and outputs its guess $r' \in \{0,1\}$ into the side-information register R'. Its other outputs ($C_{X'}$, c, and log) can remain empty, or be set to dummy values. Doing so will cause $S.\mathsf{VerDec}$ to reject, but that does not matter: $\mathcal{A}'_3$ ignores the output from $S.\mathsf{VerDec}$, and simply outputs the bit r' it received via its side-information register.

By noticing that up until the actions of $\mathcal{A}_2$ (resp. $\mathcal{A}'_2$), the two games are identical, we see that

$$\Pr[\mathsf{SymK}_{\mathcal{A},S}(\lambda) = 1] = \Pr[\mathsf{VerGame}_{\mathcal{A}',S}(\lambda) = 1]. \tag{5.17}$$

Thus, if there exists an $\mathcal{A}$ such that the left-hand probability is bounded away from $1/2$ by a nonnegligible factor, then there exists an $\mathcal{A}'$ such that the right-hand probability is, too. By contraposition, the statement of the lemma follows. □

5.4.3 Equivalence of IND-VER and SEM-VER

In this subsection, we formally show that the two verifiability concepts presented in Definitions 5.4.4 and 5.4.6 are equivalent.

Lemma 5.4.8. *If a vQHE scheme S is IND-VER, then it is SEM-VER.*

Proof. We will show this by contraposition: suppose that S is *not* SEM-VER. Then there exists a QPT $\mathcal{A}$ such that for all simulators $\mathcal{S}$, there exist QPTs $\mathcal{M}$ and $\mathcal{D}$ and a polynomial p such that the difference in acceptance probability between the real and ideal scenarios is at least $1/p(\lambda)$.

Denote the output registers of $\mathcal{A}$ as $C_{X'}$ (the ciphertext), C (the circuit), L (the computation log), and R_1' (the side-information register). Choose the simulator map $\mathcal{S}$ to be

$$\mathcal{S} : (sk, \rho^{R_1}) \mapsto \mathrm{Tr}_{X'}\left[S.\mathsf{VerDec}_{sk}^{C_{X'}CL}\left(\mathcal{A}\left(S.\mathsf{Enc}_{sk}\left(|0^n\rangle\langle 0^n|\right) \otimes \rho\right)\right)\right], \tag{5.18}$$

That is, $\mathcal{S}$ encrypts a dummy state $|0^n\rangle\langle 0^n|$, feeds it to the adversary, and checks the outcome. Note that in the accept case, the plaintext output is not the output of a correct computation, since the claimed circuit is applied to the dummy state instead of the real input. This does not matter, however, because the simulator traces out the result immediately, and only outputs the claimed circuit c and the accept/reject flag.

Since $\mathcal{S}$ is a possible simulator, there exist $\mathcal{M}$ and $\mathcal{D}$ as given by the assumption that SEM-VER is false. We construct a QPT adversary $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2', \mathcal{A}_3')$ for the VER indistinguishability game $\mathsf{VerGame}_{\mathcal{A},S}(\lambda)$ simply by setting $\mathcal{A}_1' = \mathcal{M}$ (letting the register $R = R_1R_2$), $\mathcal{A}_2' = \mathcal{A}^{C_XR_1}$, and $\mathcal{A}_3' = \mathcal{D}$. Informally, the probability that this adversary wins is

$$\Pr[r=0]\Pr[\mathcal{A}_3' \text{ guesses } 0 \mid r=0] \;+\; \Pr[r=1]\Pr[\mathcal{A}_3' \text{ guesses } 1 \mid r=1]\,. \tag{5.19}$$

More precisely, let F denote the flag register (holding acc or rej), set $\Pi_{\mathsf{acc}} = |\mathsf{acc}\rangle\langle\mathsf{acc}|$ and $\Pi_{\mathsf{rej}} = |\mathsf{rej}\rangle\langle\mathsf{rej}|$, and abbreviate

$$(c, \sigma^{XFR'}) := \mathrm{Tr}_{X'}\left[S.\mathsf{VerDec}_{sk}^{C_{X'}CL}\left(\mathcal{A}_2'^{C_XR_1}\left(S.\mathsf{Enc}_{sk}\left(|0^n\rangle\langle 0^n|\right) \otimes \mathcal{A}_1'(\rho_{evk})\right)\right)\right]. \tag{5.20}$$

Then the winning probability of $\mathcal{A}'$ is

$$\begin{aligned}&\frac{1}{2}\Pr\left[\mathcal{A}_3'\left(S.\mathsf{VerDec}_{sk}^{C_{X'}CL}\left(\mathcal{A}_2'^{C_XR_1}\left(S.\mathsf{Enc}_{sk}^{X}\left(\mathcal{A}_1'(\rho_{evk})\right)\right)\right)\right)=0\right]\\ &+\frac{1}{2}\Pr\left[\mathcal{A}_3'\left(\Phi_c^X\left(\Pi_{\mathsf{acc}}^F\sigma^{XFR'}\Pi_{\mathsf{acc}}^\dagger\right) + |\bot\rangle\langle\bot|^X \otimes \mathrm{Tr}_X\left[\Pi_{\mathsf{rej}}^F\sigma^{XFR'}\Pi_{\mathsf{rej}}^\dagger\right]\right)=1\right].\end{aligned} \tag{5.21}$$

Equation (5.21) can be verified by following the wires of Game 5.4.5 in Figure 5.3. By our definition of $\mathcal{A}'$ and $\mathcal{S}$, the probability equals

$$\frac{1}{2}\left(1-\Pr\left[\mathcal{D}\left(S.\mathsf{VerDec}_{sk}^{C_{X'}CL}\left(\mathcal{A}^{C_XR_1}\left(S.\mathsf{Enc}_{sk}^{X}\left(\mathcal{M}(\rho_{evk})\right)\right)\right)\right)=1\right]\right) \tag{5.22}$$

$$+\frac{1}{2}\Pr\left[\mathcal{D}\Big((\mathsf{ctrl}\text{-}\oslash \circ \Phi_c \circ \mathcal{S}_{sk})(\mathcal{M}(\rho_{evk}))\Big)=1\right]. \tag{5.23}$$

By the assumption that S is not SEM-VER, Equation (5.23) is $1/2$ plus a nonnegligible factor. Hence, this adversary $\mathcal{A}'$ wins the IND-VER indistinguishability game with nonnegligible advantage. □

Lemma 5.4.9. *If a vQHE scheme S is SEM-VER, then it is IND-VER.*

Proof. Suppose that a scheme S is SEM-VER, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an arbitrary QPT adversary for the IND-VER indistinguishability game. We will define a semantic adversary, message generator, and distinguisher, that together simulate the game for $\mathcal{A}$. The fact that S is SEM-VER allows us to limit the advantage of the semantic adversary over any simulator, and thereby the winning probability of $\mathcal{A}$.

By definition of SEM-VER, for $\mathcal{A}_2$ there exists $\mathcal{S}$ such that for all QPTs $\mathcal{M}$ and $\mathcal{D}$, the inequality from Definition 5.4.4 holds (with $\mathcal{A}_2$ instead of $\mathcal{A}$). We choose $\mathcal{M}$ and $\mathcal{D}$ as in Figure 5.4. More precisely, $\mathcal{M}$ does:

1. Run $\mathcal{A}_1$ on input.
2. Prepare the state $|0^n\rangle\langle 0^n|$, plus a random bit $r \in_R \{0,1\}$, and store them in the side information register R_2.
3. Swap the wires X and R_2, conditioned on r.

We also choose $\mathcal{D}$ to do the following:

1. Run $\mathcal{A}_3$ on the appropriate input wires.
2. Either apply Φ_c or $\oslash$ on the quantum state in the register R_2, conditioned on the accept/reject flag.
3. Swap the wires X' and R_2, conditioned on r.
4. Output 1 if $\mathcal{A}_3$ correctly outputs $r' = r$, and 0 otherwise.

These choices of $\mathcal{M}$ and $\mathcal{D}$ ensure that the real channel is an execution of the IND-VER game. In the ideal scenario, $\mathcal{A}_3$ receives *exactly* the same state in the cases $r = 0$ and $r = 1$. Hence, the best $\mathcal{A}_3$ can do is guess, and the probability that $r' = r$ (and thus that $\mathcal{D}$ outputs 1) is at most $1/2$.

By the assumption that S is SEM-VER, the probability that $\mathcal{D}$ outputs 1 in the real scenario can only be negligibly higher than in the ideal case. As discussed above, the real scenario corresponds exactly to the adversary $\mathcal{A}$ playing the IND-VER game. Therefore, the winning probability for $\mathcal{A}$ (i.e., the probability that $\mathsf{VerGame}_{\mathcal{A},S}(\lambda) = 1$) is at most negligibly (in λ) higher than $1/2$. □

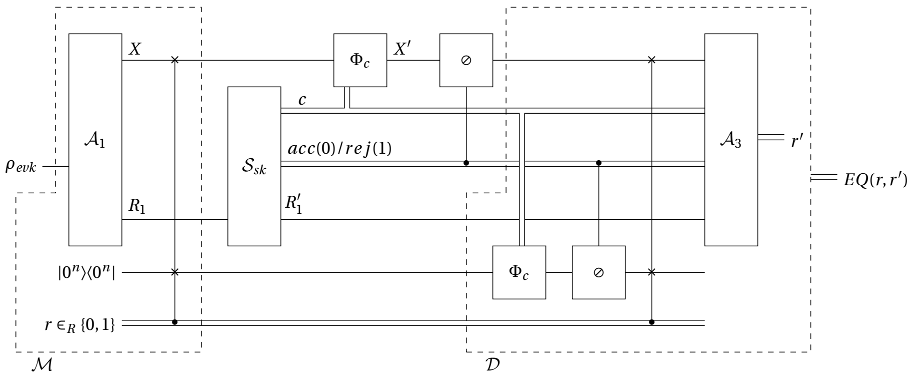


Figure 5.4: The definition of $\mathcal{M}$ and $\mathcal{D}$ in terms of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$. This figure depicts the ideal scenario in the SEM-VER definition with $\mathcal{M}$ and $\mathcal{D}$ chosen as described in the proof of Lemma 5.4.9.

5.5 TC: a noncompact partially-homomorphic scheme with verification

We now present a partially-homomorphic scheme with verification, which will serve as a building block for both the Clifford-homomorphic scheme in Section 5.6, and the fully homomorphic scheme in Section 6.4. It is called TC (for "trap code"), and is homomorphic only for CNOT, (classically controlled) Paulis, and measurement in the computational and Hadamard basis. It does not satisfy compactness: as such, it performs worse than the trivial scheme where the client performs the circuit at decryption time. However, TC lays the groundwork for the vQHE schemes we present in Sections 5.6 and 6.4, and is therefore important to understand in detail. It is a variant of the trap code [BGS13] (which requires classical interaction for T gates), adapted to our vQHE framework. For simplicity we consider the regular trap code in this chapter, but the scheme can easily be modified to work with the strong trap code.

Key generation and encryption

Let CSS be a (publicly known) self-dual $[[n,1,d]]$ CSS code, so that H and CNOT are transversal. We choose $d = 2\lambda + 1$. The code needs to satisfy $n = \text{poly}(\lambda)$: the concatenated Steane code satisfies this relationship, as does the Reed–Muller code presented in Section 3.7.3.

For a circuit with m input wires, we generate keys for the trap code as follows. Choose a single random permutation $\pi \in_R S_{3n}$. For each qubit $i \in \{1,\ldots,m\}$, sample $x[i] \in_R \{0,1\}^{3n}$ and $z[i] \in_R \{0,1\}^{3n}$. The secret key sk is $(\pi, x[1], z[1], \ldots, x[n], z[n])$, and

ρ_{evk} is left empty.

Encryption of the ith qubit is the trap-code encoding using the permutation π and Pauli key $x[i], z[i]$. We denote the encoding procedure as TC.Enc, and write $\widetilde{\sigma}$ for an encrypted version of a state σ.

Evaluation

For a more detailed explanation on how to compute on trap-code encoded qubits, refer to Section 3.4.2. Here, we give a brief overview of how those techniques fit within the setting of vQHE.

The application of Pauli gates (X and/or Z) can be achieved without touching the actual state, by updating the keys to the quantum one-time pad in the appropriate way. This is a classical task, so we can postpone it to TC.VerDec. (Recall that TC.VerDec gets the circuit description.) So, formally, the evaluation procedure for Pauli gates is the identity map. Paulis conditioned on a classical bit b known to TC.VerDec at execution time (e.g., a measurement outcome) can be applied in the same manner.

To apply a CNOT to encrypted qubits σ_i and σ_j, we apply CNOT transversally between the $3n$ qubits of $\widetilde{\sigma_i}$ and the $3n$ qubits of $\widetilde{\sigma_j}$. Ignoring the quantum one-time pad for the moment, the effect is a logical application of CNOT on $\sigma_i \otimes \sigma_j$. The quantum one-time pad requires a key update, which happens in TC.VerDec.

Computational-basis measurement is performed by measuring all $3n$ physical qubits. During TC.VerDec, the contents of the measured qubits (now a classical string $\widetilde{w} \in \{0,1\}^{3n}$) will be interpreted into a logical measurement outcome.

Finally, we handle Hadamard-basis measurements. A transversal application of H to all $3n$ relevant physical qubits precedes the evaluation procedure for a computational-basis measurement. Since CSS is self-dual, this action applies a logical H. Since $\mathsf{H}|0\rangle = |+\rangle$ and $\mathsf{H}|+\rangle = |0\rangle$, all computational traps are swapped with the Hadamard traps. This is reflected in the way TC.VerDec checks the traps (see Protocol 5.5.2).

Verification and decryption

If a qubit is unmeasured after evaluation (because it is not measured in the circuit), then TC.VerDecQubit is applied: this is the regular decoding procedure of the trap code which, given a key (π, x, z), undoes the permutation, checks all traps against he keys x and z, and decodes the CSS code.

If a qubit is measured during evaluation, TC.VerDec receives a list $\widetilde{w}$ of $3n$ physical measurement outcomes for that qubit. These outcomes are classically processed to produce the plaintext measurement outcome:

Protocol 5.5.1 (TC.VerDecMeasurement)**.** Given a secret key (π, x, z), a measurement string $\widetilde{w} \in \{0,1\}^{3n}$, and a basis *basis* $\in \{+, \times\}$, do the following:

1. If *basis* = +, let $w' := \pi^{-1}(x \oplus \widetilde{w})$, Check the second n bits of w': if they are not all zero, return (0, rej).
2. If *basis* = ×, let $w' := \pi^{-1}(z \oplus \widetilde{w})$, Check the third n bits of w': if they are not all zero, return (0, rej).
3. If the previous steps did not reject, compute CSS.ClassicalDec on the first n bits of w', and return its result (both the decryption value and the accept/reject flag).

Here, CSS.ClassicalDec is the classical decoding procedure that recovers the logical measurement outcome from the physical ones (see Section 3.4.2): therefore, TC.VerDecMeasurement is also completely classical. Note that we only check the $|0\rangle$ traps for a computational-basis measurement. Intuitively, this does not affect security, since any attack that affects only $|+\rangle$ but not $|0\rangle$ will be canceled by the measurement. A similar reasoning applies for a Hadamard-basis measurement.

The complete procedure TC.VerDec updates the quantum-one-time-pad keys according to the gates in the circuit description, and then decrypts all qubits and measurements as described above.

Protocol 5.5.2 (TC.VerDec)**.** Given a secret key (π, x, z), an output ciphertext $\widetilde{\sigma}$ (some qubits $\widetilde{\sigma_i}$ may be interpreted as classical measurement strings), and a circuit c, do the following:

1. For all gates G in the circuit c, do the following:

   If $G = \mathsf{X}_i$**:** update $x[i] \leftarrow x[i] \oplus \pi(1^n 0^{2n})$.

   If $G = \mathsf{Z}_i$**:** update $z[i] \leftarrow z[i] \oplus \pi(1^n 0^{2n})$.

   If $G = \mathsf{CNOT}_{ij}$**:** update $(x[i], z[i]), (x[j], z[j]) \leftarrow (x[i], z[i] \oplus z[j]), (x[i] \oplus x[j], z[j])$.

   If G **is a measurement in basis** *basis* **on qubit** i**:** Execute Protocol 5.5.1 to get $(a_i, \mathit{flag}) \leftarrow \mathsf{TC.VerDecMeasurement}((\pi, x[i], z[i]), \widetilde{\sigma_i}, \mathit{basis})$. If *flag* = rej, return $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$. Otherwise, continue with the next gate.

2. Execute TC.VerDecQubit (the regular trap-code decoding) on all unmeasured qubits using the updated keys. If one rejects, return $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
3. Trace out all wires that are not part of the output of c (e.g., because they were auxiliary wires), and return the remaining list of decoded qubits and measurement outcomes, along with the flag $|\mathsf{acc}\rangle\langle\mathsf{acc}|$.

For CNOT, the key update is motivated by Equations (2.10) to (2.13) that describe the commutation relations between CNOT and the Pauli group. Since both qubit i and j are permuted in the same way, updating the key as described in Protocol 5.5.2 correctly deals with commuting the CNOT through the quantum one-time pad.

5.5.1 Correctness, privacy, and noncompactness

After an honest evaluation, TC.VerDec accepts with probability 1. This correctness is straightforward to check by following the evaluation (and decryption) gate-by-gate.

For privacy, since we will show that TC is IND-VER (in the next section), it follows from Lemma 5.4.7 that it is also q-IND. However, note that the final step in the encryption procedure is the application of a information-theoretically secure quantum one-time pad with fresh, independent keys. Thus, the privacy of TC does not rely on computational assumptions.

TC is not compact in the sense of Definition 5.4.3, however. In order to compute the final decryption keys, the whole gate-by-gate key update procedure needs to be executed, aided by the computation log and information about the circuit. Thus, we cannot break TC.VerDec up into two separate functionalities, Ver and Dec, where Dec can successfully retrieve the keys and decrypt the state, based on only the output ciphertext and the secret key.

5.5.2 Secure verifiability

We already know that the trap code provides authentication of quantum states [BW16; BGS13] (see also Section 3.3.2). We use similar strategies to prove IND-VER for TC:

Theorem 5.5.3. TC *is a IND-VER secure (somewhat-)homomorphic-encryption scheme. That is, for any adversary $\mathcal{A}$,*

$$\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TC}}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary for TC, for the $\mathsf{VerGame}_{\mathcal{A},\mathsf{TC}}(\lambda)$ security game (see Figure 5.3). Let $sk = (\pi, x, z)$ be a uniformly random key, with $\pi \in S_{3n}$, $x, z \in \{0,1\}^{3mn}$ (where m is the number of input qubits). Let CSS be the underlying $[[n,1,d]]$ CSS code, with $d = 2\lambda + 1$.

We argue that the probability that TC.VerDec accepts in $\mathsf{VerGame}_{\mathcal{A},\mathsf{TC}}(\lambda)$ is independent of r, and that in the reject case, $\mathcal{A}$ has no advantage in guessing r. These two facts allow us to focus our attention to the accept case only, by showing that $\mathcal{A}$ does not have significant advantage in guessing r in that case either.

To see that the accept probability does not depend on r, first note that the choice of the circuit c (output by $\mathcal{A}_2$ in register C) cannot depend on the bit r chosen by the challenger. The output of TC.Enc looks fully mixed regardless of the value of r, due to the quantum one-time pad. Furthermore, note that the decryption procedure only considers the tag qubits when deciding whether to accept or reject, plus the claimed circuit c (which does not depend on r). In particular, it does not depend on the data qubits, which *do* depend on r. In fact, we can imagine delaying undoing the QOTP on those data qubits until after the accept or reject choice, leaving r fully hidden at the time the decision to accept or reject is made.

In the reject case, TC.VerDec outputs a fixed quantum state $|\bot\rangle\langle\bot|$, and will never reveal the QOTP key that is applied to the input of TC.Enc. Thus, in that case, $\mathcal{A}_3$ cannot do better than a random guess for r.

For the rest of this proof, we consider only the accept case, and show that even then, $\mathcal{A}$ has a negligible advantage in guessing r.

For each logical qubit $i \in [m]$, let $B_i^c \in \{\mathsf{id}, \mathsf{comp}, \mathsf{had}\}$ describe whether the qubit is supposed to be unmeasured, measured in the computational basis, or measured in the Hadamard basis, respectively, in the circuit c. Define $M_{\mathsf{id}} = \mathbb{I}_{2^n} \otimes |0\rangle\langle 0|^{\otimes n} \otimes |+\rangle\langle +|^{\otimes n}$ as the projector corresponding to accepting the traps of an unmeasured qubit (after undoing the permutation and one-time pad). Similarly define $M_{\mathsf{comp}} = \mathbb{I}_{2^n} \otimes |0\rangle\langle 0|^{\otimes n} \otimes \mathbb{I}_{2^n}$ and $M_{\mathsf{had}} = \mathbb{I}_{2^n} \otimes \mathbb{I}_{2^n} \otimes |+\rangle\langle +|^{\otimes n}$ as the projectors corresponding to accepting the traps of a measured qubit. For $B^c := (B_1^c, B_2^c, \ldots, B_m^c)$, write $M_{B^c} := \bigotimes_{i\in[m]} M_{B_i^c}$.

To simplify notation, we assume without loss of generality that $\mathcal{H}_R = \mathcal{H}_{R'}$, and that $\mathcal{A}_2$ is a unitary [2] acting on registers C_X (the ciphertext), R (the side information), C (the circuit, initialized to the all-zero state) and L (the log, initialized to the all-zero state). Since the log is left empty in TC, we will ignore it for this proof. Again without loss of generality, we can assume that $\mathcal{A}_2$ consists of two parts: an arbitrary attack unitary $U^{C_X CR}$, followed by an "honest" unitary $D^{C_X C}$, consisting of the quantum operation prescribed by TC.Eval for the circuit in the C register: transversal CNOT gates on the encrypted states in C_X. We do not include the honest measurement actions in $\mathcal{A}_2$, but instead postpone them to TC.VerDec.

We would like to apply the Pauli twirl (Lemma 2.4.5) in order to break up the attack unitary U into a mixture of Pauli attacks. However, TC.VerDecMeasurement prevents us from doing so: it does not completely undo the one-time pad on the measured qubits (e.g., for a computational-basis measurement, the Z keys are not undone). To enable the use of the Pauli twirl, we use the fact that for the trap code, measure-then-decode is equivalent to decode-then-measure [BGS13]: instead of the honest measurement followed by a classical decoding of the measurement result, we can quantumly decode the state and then measure it. In the accept case, these two approaches are completely equivalent.

Let $\sigma \in \mathcal{H}_{XR}$ be the output state of $\mathcal{A}_1$, and define

$$\rho^{C_X R} := \mathsf{CSS.Enc}_{sk}^{X}(\sigma) \otimes |0\rangle\langle 0|^{\otimes n} \otimes |+\rangle\langle +|^{\otimes n} \tag{5.24}$$

to be the state after the CSS encoding, but before the permutation and one-time pad: that is, the state $(\mathsf{X}^x \mathsf{Z}^z \pi^{\otimes m})^{C_X} \rho (\mathsf{X}^x \mathsf{Z}^z \pi^{\otimes m})^\dagger$ is the encrypted input to $\mathcal{A}_2$ in case

[2]We can expand $\mathcal{H}_R$ to a bigger space to achieve unitarity of $\mathcal{A}_2$, if necessary.

$r = 0$, together with a circuit register that is initialized to the all-zero state $|0\rangle\langle 0|^C$ of unspecified size polynomial in m.

Furthermore, let $F_{\pi,x,z} : |\psi\rangle^{C_X} |c\rangle^C \mapsto \left(\pi^{\dagger\otimes m}\mathsf{X}^{f_c(x,z,\pi)}\mathsf{Z}^{f'_c(x,z,\pi)}|\psi\rangle\right)|c\rangle$ be the unitary map that reads a circuit description c, updates the keys x and z according to that circuit, and removes the quantum one-time pad and permutations. Essentially, it is TC.VerDec without the CSS.Dec and measurement steps. Note that $F_{\pi,x,z}$ can be written as $F'\left(\pi^{\dagger\otimes m} \otimes \mathbb{I}^C\right) D \left(\mathsf{X}^x\mathsf{Z}^z \otimes \mathbb{I}^C\right) D^\dagger$, where $F'^{C_X C}$ is a Pauli correction (conditioned on the register C) that does not depend on π, x, z. It represents the Pauli gates that occur in the circuit, applied transversally to the first n qubits of each ciphertext, and can be delayed until the end of the decryption by commuting them through the CNOT operations in the honest unitary D, and the permutations.

The plaintext that TC.VerDec holds right before running CSS.Dec, projected to the accept case, equals

$$M_{B^c} \mathop{\mathbb{E}}_{\substack{\pi\in S_{3n}\\ x,z\in\{0,1\}^{3mn}}} \left[F_{\pi,x,z}^{C_X C} D^{C_X C} U^{C_X C R}\left(\mathsf{X}^x\mathsf{Z}^z\pi^{\otimes m}\right)^{C_X} \rho\pi^{\dagger\otimes m}\mathsf{X}^x Z^z U^\dagger D^\dagger F_{\pi,x,z}^\dagger\right] \tag{5.25}$$

$$= M_{B^c} F' \mathop{\mathbb{E}}_{\substack{\pi\in S_{3n}\\ x,z\in\{0,1\}^{3mn}}} \left[\pi^{\otimes m} D\mathsf{X}^x\mathsf{Z}^z U\mathsf{X}^x\mathsf{Z}^z\pi^{\otimes m}\rho\pi^{\dagger\otimes m}\mathsf{X}^x Z^z U^\dagger \mathsf{X}^x\mathsf{Z}^z D^\dagger\pi^{\otimes m}\right] F'^\dagger \tag{5.26}$$

$$= M_{B^c} F' D \mathop{\mathbb{E}}_{\substack{\pi\in S_{3n}\\ x,z\in\{0,1\}^{3mn}}} \left[\pi^{\otimes m}\mathsf{X}^x\mathsf{Z}^z U\mathsf{X}^x\mathsf{Z}^z\pi^{\otimes m}\rho\pi^{\dagger\otimes m}\mathsf{X}^x Z^z U^\dagger \mathsf{X}^x\mathsf{Z}^z\pi^{\otimes m}\right] D^\dagger F'^\dagger. \tag{5.27}$$

The last equality is due to the fact that $\pi^{\otimes m}D = D\pi^{\otimes m}$, since the honest action D applies CNOT operations transversally on all $3n + 3n$ physical qubits whenever a logical CNOT appears in the circuit. This transversal operation is invariant under the permutation π.

We are finally able to apply the Pauli twirl (Lemma 2.4.5), because other than the $\mathsf{X}^x\mathsf{Z}^z$, no other terms depend on x and z. Write $U = \sum_{P\in\mathscr{P}_{3mn}} \alpha_P P^{C_X} U_P^{CR}$. Then we can decompose U into a probabilistic mixture of Pauli operations, rewriting Equation (5.27) as

$$M_{B^c} F' D \mathop{\mathbb{E}}_{\substack{\pi\in S_{3n}\\ P\in\mathscr{P}_{3mn}}} \left[|\alpha_P|^2\pi^{\otimes m}(P\otimes U_P)\pi^{\otimes m}\rho\pi^{\dagger\otimes m}(P\otimes U_P^\dagger)\pi^{\otimes m}\right] D^\dagger F'^\dagger. \tag{5.28}$$

Expressions of this form have been carefully analyzed in earlier trap-code security proofs: see, e.g., the analysis of Broadbent and Wainewright [BW16]. For completeness, we will sketch the analysis here. The goal is to argue that Equation (5.28) is negligibly close (in λ) to the state at that point in the game if $r = 1$.

We start by noting that $F'D$ represents the application of the circuit stored in C to the state ρ: namely, it applies CNOT gates transversally, followed by a Pauli that represents the Paulis in the circuit, commuted through the CNOT operations and applied transversally to the first n qubits of each ciphertext.

Whether or not the state in Equation (5.28) is equal to the $r = 1$ case (where the input is swapped out, the circuit is applied, and it is swapped back in) depends on whether or not the Pauli operators $P = \bigotimes_i P_i$ alter any logical qubit i. The permutation π ensures that the attacker cannot choose on which locations the nonidentity terms in P_i act, so they essentially only choose how many X, Y, and Z terms the operator P_i contains.

Let us first consider an unmeasured qubit i. Because CSS can correct up to λ errors (since $d = 2\lambda + 1$), only those Paulis P_i that are nonidentity in more than λ locations will cause the logical qubit after decoding to change. For such Pauli terms, assume without loss of generality that P_i has X-weight at least $\lambda/2$ out of the $3n$ physical qubits that encode the ith qubit.[3] The probability over π that *none* of the X components ends up on the computational-basis traps (i.e., positions $m+1$ to $2m$) is upper bounded by $(2/3)^{\lambda/2}$, since each individual X operation has probability $2/3$ of missing the computational-basis traps. (A more careful combinatorial analysis including the Z operations can improve the bound to $(2/3)^{\lambda}$ [BW16], but this simple bound suffices for us.)

Next, consider a qubit i that is measured in the computational basis (the argument for Hadamard-basis measurements is similar). For these qubits, the Z components of the attack are not detected, but they also do not change the output: since the data qubits are also measured, only the X Paulis will change the data contents. Therefore, the operator P_i will have to contain at least λ Pauli X operators on the $3n$ physical qubits. Repeating the same argument as for the unmeasured qubits, we see that the probability over π that all computational-basis traps are missed is at most $(2/3)^{\lambda}$.

The above analysis shows that a Pauli operator P_i either does not have enough weight to affect the logical outcome (for those terms, the cases $r = 0$ and $r = 1$ are identical), or it only has very small probability of being accepted, and therefore very small norm in Equation (5.28). Hence, the trace distance between the states that $\mathcal{A}_3$ receives in the $r = 0$ and $r = 1$ case is negligible. The final guessing probability is bounded as

$$\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TC}}(\lambda) = 1] \leqslant \frac{1}{2} + \frac{1}{2}\left(\frac{2}{3}\right)^{\lambda}, \tag{5.29}$$

so that the guessing advantage is negligible in λ. □

In Section 5.6, we will use the IND-VER property of TC to prove verifiability for our new scheme. In order to achieve verifiability for that scheme, we will actually need a slightly stronger notion of verifiability for TC: IND-VER-n, where the adversary is allowed to submit plaintexts in n rounds, which are either all encrypted or all swapped out. In this subsection, we show that TC also fulfills this stronger notion. For our purposes in Section 5.6, it suffices to show that TC is secure against an adversary that

[3]For operators with more Z components than X components, we could argue using Z instead.

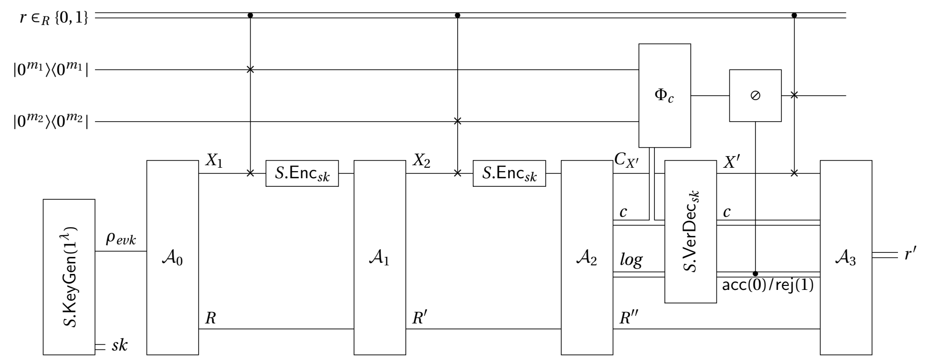


Figure 5.5: The game $\mathsf{VerGame}^2_{\mathcal{A},S}(\lambda)$. The integers m_1 and m_2 are the respective sizes of the X_1 and X_2 registers.

is allowed two rounds (IND-VER-2), but the definitions and proof trivially extend to the general case.

Definition 5.5.4 (IND-VER-2 game)**.** For an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a scheme S, and a security parameter λ, $\mathsf{VerGame}^2_{\mathcal{A},S}(\lambda)$ is shown in Figure 5.5. It is identical to $\mathsf{VerGame}_{\mathcal{A},S}(\lambda)$, except that the adversary gets to submit two plaintexts for encryption, which are both encrypted with the same permutation, but fresh quantum one-time pads. The circuit c may act on both encrypted inputs.

Definition 5.5.5 (IND-VER-2)**.** A vQHE scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{VerDec})$ has *2-round λ-indistinguishable verification* (IND-VER-2) if for any QPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$,

$$\Pr[\mathsf{VerGame}^2_{\mathcal{A},S}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Here, the probability is taken over $\mathsf{KeyGen}(1^\lambda), \mathsf{Enc}, \mathsf{VerDec}$, and $\mathcal{A}$.

Lemma 5.5.6. TC *is IND-VER-2.*

Proof. Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an arbitrary polynomial-time adversary for the VER-2 indistinguishability game for TC. For notational convenience, write the secret key as $sk = (\pi, x_1, z_1, x_2, z_2)$, where x_1 and z_1 are lists of $3nm_1$ bits, sufficient for encrypting X_1, and analogously x_2 and z_2 are lists of $3nm_2$ bits.

We now slightly alter the VER-2 game in the following way. In the first encryption step of the game, instead of providing $\mathcal{A}_1$ with $\mathsf{TC.Enc}_{(\pi,x_1,z_1)}$ applied to the register X_1, we provide $\mathcal{A}_1$ with halves of m_1 EPR pairs, and perform Bell measurements between the other halves and the qubits in X_1, after they have been encrypted with $\mathsf{TC.Enc}_{(\pi,x_1,z_1)}$. Let the outcomes of these measurements be given by $a, b \in \{0,1\}^{3nm_1}$:

a and b describe the effective X and Z Paulis that are applied to X_1 by these teleportation measurements. To undo these Paulis, we update sk to $(\pi, x_1 \oplus a, z_1 \oplus b, x_2, z_2)$ at this point. Since the quantum one-time pad keys x_1 and z_1 are chosen uniformly at random, and are completely hidden from the perspective of the adversary, the new keys $x_1 \oplus a$ and $z_1 \oplus b$ are valid keys that are sampled from the same distribution. Hence, the winning probability of $\mathcal{A}$ is not affected by this change of the game.

A second small change to the game is the following: instead of performing the Bell measurements and the secret-key update immediately, it is done only after $\mathcal{A}_1$ has provided its query in X_2. Since these actions happen only on wires which are not accessible to $\mathcal{A}_1$ (and otherwise also not touched in this stage of the game), this change also does not affect the execution or outcome of the game in any way.

We have now arrived at an interesting situation: $\mathcal{A}_1$ only receives halves of EPR pairs, and so its choice for X_2 or R' is not based on the first ciphertext received from the challenger – that ciphertext will only be generated after execution of $\mathcal{A}_1$. We can merge $\mathcal{A}_0$ and $\mathcal{A}_1$ into a single QPT algorithm that produces X_1 and X_2 simultaneously. When viewed as such, $\mathcal{A}$ is an adversary for the single-query VER indistinguishability game, and we can conclude that

$$\Pr[\mathsf{VerGame}^2_{\mathcal{A},\mathsf{TC}}(\lambda) = 1] = \Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TC}}(\lambda) = 1]. \tag{5.30}$$

Since we know that the latter probability is bounded by $1/2 + \text{negl}(\lambda)$ from Theorem 5.5.3, so is the first. □

Lemma 5.5.6, and its generalization to IND-VER-n, may be seen as an argument that TC is also q-IND-CPA secure. After all, in q-IND-CPA, the adversary is supplied with an encryption oracle which it can query before submitting its challenge plaintext. In IND-VER-n, we do the same, except we make it easier to tell the difference between $r = 0$ and $r = 1$ by having *all* encryption queries depend on r. However, the crucial difference between IND-VER-n and q-IND-CPA is that in the former, (part of) the secret key is resampled for every encryption. q-IND-CPA requires the encryptions to use the same secret key for every query, and for the challenge encryption. TC does therefore not provide q-IND-CPA security out of the box. If q-IND-CPA security is desired, one could easily extend TC by using a pseudorandom function for the quantum one-time pad [ABF+16]. Alternatively, TC can be adapted by sampling the one-time-pad keys freshly every time, during encryption: we adopt this strategy for later schemes. Both adaptations of TC require computational assumptions.

5.6 TCL: a partially-homomorphic scheme with verification

In this section, we introduce an encryption scheme TCL that is homomorphic for all Clifford operations. In contrast to CL, it is verifiable, and in contrast to TC, it is compact. The scheme TCL will be *leveled* (see Section 5.2.1), because the evaluation of a phase gate P and a Hadamard gate H will consume an encoded magic state from the evaluation key. The number of magic states in the evaluation key gives an upper bound to the number of P and H gates in the evaluated circuit.

The scheme TCL is very similar to TC, but with a few additions. First, the key generation procedure will be given parameters p and h, which indicate the amount of encoded magic states to be produced. Second, to ensure compactness, one-time-pad key updates will be performed during evaluation rather than decryption. To accommodate the key updates, we encrypt the pad keys under a classical homomorphic scheme HE, just as in the unverifiable Clifford scheme CL. The evaluator writes all classical homomorphic computations in the computation log, so that the verification procedure may check that they were performed honestly. However, we need to ensure that the computation log *starts* with the honest encryption of the one-time-pad keys (and that the evaluator did not secretly apply an extra gate before starting the official log). To this end, we authenticate the initial encryptions of the pad keys using a classical message authentication code.

Let $\lambda \in \mathbb{N}$ be a security parameter, and let $p, h \in \mathbb{N}$ be an upper bound on the number of P and H gates (respectively) that will be in the evaluated circuit. Fix a classical message authentication code $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{Ver})$ that is existentially unforgeable under adaptive chosen message attacks (EUF-CMA [KL14]) by a quantum adversary. For example, one may take the standard pseudorandom-function construction [KL14] with a post-quantum PRF. The Tag procedure defines an authentication procedure $\mathsf{MAC.Sign}_k : m \mapsto (m, \mathsf{MAC.Tag}_k(m))$. Note that, contrary to the quantum setting, the classical authentication code does not provide privacy by itself.

Let CSS be the same $[[n, 1, d]]$ CSS code as in Section 5.5. That is, $n = \text{poly}(\lambda)$, $d = 2\lambda + 1$, and the gates H and CNOT are transversal.

The evaluation key will contain a number of auxiliary states for the P and H gates (see also Section 3.4.2). Those states are encrypted using the same "global" permutation π, but with fresh one-time pad keys.

Protocol 5.6.1 (TCL.KeyGen)**.** Given a security parameter λ and gate parameters p, h, do the following:

1. Sample classical keys as

$$k \leftarrow \mathsf{MAC.KeyGen}, \tag{5.31}$$
$$\pi \leftarrow S_{3n}, \tag{5.32}$$
$$(sk, pk, evk) \leftarrow \mathsf{HE.KeyGen}. \tag{5.33}$$

2. Set (π, k, sk, pk) to be the secret key.

3. For $i \in [p]$, let $\mu_i^{\mathsf{P}} \leftarrow \mathsf{TCL.Enc}_{(\pi,k,sk,pk)}(\mathsf{P}\,|+\rangle)$ be the encrypted magic state for P. For $\mathsf{TCL.Enc}$, see Protocol 5.6.2 below.
4. For $i \in [h]$, let $\mu_i^{\mathsf{H}} \leftarrow \mathsf{TCL.Enc}_{(\pi,k,sk,pk)}\left(\frac{1}{\sqrt{2}}(\mathsf{H} \otimes \mathsf{I})\,(|00\rangle + |11\rangle)\right)$ be the encrypted magic state for H.
5. Let $keys \leftarrow \mathsf{MAC.Sign}_k(pk, evk, \mathsf{HE.Enc}_{pk}(\pi))$ be the authenticated list of keys. The secret information π is encrypted before it is signed.
6. Output the secret key (π, k, sk, pk), along with the quantum evaluation key $(keys, \mu_1^{\mathsf{P}}, \mu_2^{\mathsf{P}}, \dots, \mu_p^{\mathsf{P}}, \mu_1^{\mathsf{H}}, \mu_2^{\mathsf{H}}, \dots, \mu_h^{\mathsf{H}})$.

Perhaps somewhat counter-intuitively, the secret key contains the classical public key pk. This is merely because pk is required for encryption, and we have no specific intention of keeping the information pk private.

The encryption of a quantum state is a combination of $\mathsf{TC.Enc}$ and $\mathsf{CL.Enc}$: the state is encoded in the trap code, but the keys to the one-time pad are chosen during encryption (rather than during key generation), and appended to the ciphertext in an encrypted and authenticated form.

Protocol 5.6.2 ($\mathsf{TCL.Enc}$)**.** Given the secret key (π, k, sk, pk) and the quantum plaintext σ, output the state

$$\widetilde{\sigma} := \sum_{x,z \in \{0,1\}^{3n}} \left(\mathsf{TC.Enc}_{\pi,x,z}(\sigma) \otimes \mathsf{MAC.Sign}_k\left(\mathsf{HE.Enc}_{pk}(x,z)\right)\right). \tag{5.34}$$

Evaluation of gates is analogous to the interactive evaluation scheme using the trap code [BGS13], briefly discussed in Section 3.4.2, except that the interactions are replaced by classical homomorphic evaluation.

Evaluation of a circuit c is done gate-by-gate, as described by the algorithms below. We will assume throughout that $\mathsf{HE.Enc}$ and $\mathsf{HE.Eval}$ produce, apart from their actual output, a complete *computation log* describing all classical randomness used, all classical computation steps, and all intermediate classical results.

Measurement. Recall that $\mathsf{TC.VerDecMeasurement}$ is a completely classical procedure that decodes the list of $3n$ measurement outcomes into the logical outcome, and checks the relevant traps. A computational-basis measurement in TCL is then defined as:

Protocol 5.6.3 ($\mathsf{TCL.EvalMeasurement}$)**.** Given a ciphertext $\widetilde{\sigma}$ (with encrypted one-time pad keys $\widetilde{x}$ and $\widetilde{z}$) and an evaluation key containing $\widetilde{\pi}$, pk, and evk, do the following:

- Measure the qubits of $\widetilde{\sigma}$ in the computational basis. Call the result $a \in \{0,1\}^{3n}$.
- Let $(\widetilde{a}, log_1) \leftarrow \mathsf{HE.Enc}_{pk}(a)$.
- Let $(\widetilde{b}, \widetilde{flag}, log_2) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{TC.VerDecMeasurement}}((\widetilde{\pi}, \widetilde{x}, \widetilde{z}), \widetilde{a}, \mathsf{HE.Enc}_{pk}(+))$. The (encrypted) bit b represents the logical measurement outcome, and the flag is either accept or reject.
- Output $\widetilde{b}, \widetilde{flag}, (log_1, log_2)$.

Essentially, $\mathsf{TCL.EvalMeasurement}$ performs both the server-side and client-side computation of the TC measurement protocol. The client-side computations are classical, and can be performed homomorphically.

Hadamard-basis measurement is performed similarly, except the qubits of $\widetilde{\sigma}$ are measured in the Hadamard basis and $\mathsf{HE.Enc}_{pk}(\times)$ is given as the last argument for the evaluation of $\mathsf{TC.VerDecMeasurement}$.

Pauli gates. A logical Pauli-X is performed by (homomorphically) flipping the X-key bits of the QOTP, as in TC. The (classically controlled) evaluation of a Pauli-Z works the same way, only the relevant bits in $\widetilde{z}$ are flipped.

Protocol 5.6.4 ($\mathsf{TCL.Eval}^{\mathsf{X}}$)**.** Given a ciphertext $\widetilde{\sigma}$ (with encrypted one-time pad keys $\widetilde{x}$ and $\widetilde{z}$) and an evaluation key containing $\widetilde{\pi}$, pk, and evk, do the following:

1. Homomorphically update the key $\widetilde{x}$ in three steps:
   (a) Let $(\widetilde{y}, log_1) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{unpermute}}(\widetilde{\pi}, \widetilde{x})$. Here, $\mathsf{unpermute}$ is the function that takes a permutation π and a string x as input, and outputs $\pi^{-1}(x)$.
   (b) Let $(\widetilde{y'}, log_2) \leftarrow \mathsf{HE.Eval}_{evk}^{\oplus}(\widetilde{y}, \mathsf{HE.Enc}_{pk}(1^n 0^{2n}))$. This operation flips the first n bits, containing the data.
   (c) Let $(\widetilde{x'}, log_3) \leftarrow \mathsf{HE.Eval}_{evk}^{\mathsf{permute}}(\widetilde{\pi}, \widetilde{y'})$.
2. Output $\widetilde{\sigma}$ (with $\widetilde{x}$ replaced by $\widetilde{x'}$) and (log_1, log_2, log_3).

Since the key-update operation is classical, the functionality extends straightforwardly to a classically controlled Pauli-X (by specifying an additional bit b encrypted into $\widetilde{b}$ that indicates whether or not X should be applied):

Protocol 5.6.5 ($\mathsf{TCL.Eval}^{\mathsf{CondX}}$)**.** Given a ciphertext $\widetilde{\sigma}$ (with encrypted one-time pad keys $\widetilde{x}$ and $\widetilde{z}$) and an evaluation key containing $\widetilde{\pi}$, *pk*, and *evk*, do the following:

1. Homomorphically update the key $\widetilde{x}$ in four steps:
   (a) Let $(\widetilde{y}, log_1) \leftarrow \mathsf{HE.Eval}^{\mathsf{unpermute}}_{evk}(\widetilde{\pi}, \widetilde{x})$. Here $\mathsf{unpermute}$ is the function that takes a permutation π and a string x as input, and outputs $\pi^{-1}(x)$.
   (b) Let $\widetilde{s} \leftarrow \mathsf{HE.Eval}^{z \mapsto z^n 0^{2n}}_{evk}(\widetilde{b})$.
   (c) Let $(\widetilde{y'}, log_2) \leftarrow \mathsf{HE.Eval}^{\oplus}_{evk}(\widetilde{y}, \widetilde{s})$. This operation flips the first n bits, but only if $b = 1$.
   (d) Let $(\widetilde{x'}, log_3) \leftarrow \mathsf{HE.Eval}^{\mathsf{permute}}_{evk}(\widetilde{\pi}, \widetilde{y'})$.
2. Output $\widetilde{\sigma}$ (with $\widetilde{x}$ replaced by $\widetilde{x'}$) and (log_1, log_2, log_3).

CNOT gates. The evaluation of CNOT in TCL is analogous to TC, only the key updates are performed homomorphically during evaluation.

Protocol 5.6.6 ($\mathsf{TCL.Eval}^{\mathsf{CNOT}}$)**.** Given ciphertexts $\widetilde{\sigma_1}$ and $\widetilde{\sigma_2}$ (with encrypted one-time pad keys $\widetilde{x_1}, \widetilde{z_1}, \widetilde{x_2}, \widetilde{z_2}$) and an evaluation key containing $\widetilde{\pi}$, *pk*, and *evk*, do the following:

1. Apply CNOT transversally, each time between the ith qubit in $\widetilde{\sigma_1}$ and the ith qubit in $\widetilde{\sigma_2}$.
2. Homomorphically update the keys according to the commutation rule in Equation (5.8). That is, let
$$(\widetilde{x'_1}, \widetilde{x'_2}, \widetilde{z'_1}, \widetilde{z'_2}, log) \leftarrow \mathsf{HE.Eval}^{\mathsf{CNOT}\text{-update}}_{evk}(\widetilde{x_1}, \widetilde{x_2}, \widetilde{z_1}, \widetilde{z_2}). \tag{5.35}$$
3. Output σ_1, σ_2 (with $\widetilde{x_1}, \widetilde{z_1}, \widetilde{x_2}, \widetilde{z_2}$ replaced by $\widetilde{x'_1}, \widetilde{x'_2}, \widetilde{z'_1}, \widetilde{z'_2}$) and *log*.

Phase gates. Performing a P gate requires homomorphic evaluation of all the above gates: (classically controlled) Paulis, CNOTs, and measurements. We also consume the state μ_i^{P} (an encryption of the magic state $\mathsf{P}\,|+\rangle$) for the ith phase gate in the circuit. We define $\mathsf{TCL.Eval}^{\mathsf{P}}$ to be the concatenation of the evaluations corresponding to the gates in the magic-state-computation circuit from Equation (3.25). The overall computation log is just a concatenation of the logs.

Hadamard gates. The Hadamard gate can be performed in the same manner as the phase gate. The ith gate consumes μ_i^{H}, an encryption of the magic state $(\mathsf{H} \otimes \mathsf{I}) |\Phi^+\rangle$. See the circuit from Equation (3.26).

The decryption procedure for TCL consists of two parts. First, we perform several classical checks. This includes MAC-verification of all classically authenticated messages, and checking that the gates listed in the log match the circuit description. We also check the portions of the log which specify the (purely classical, FHE) steps taken during HE.Enc and HE.Eval; we call this last step TCL.CheckLog. Secondly, we check all unmeasured traps and decode the remaining qubits, just as in TC. We reject if TCL.CheckLog rejects, or if the traps have been triggered.

Protocol 5.6.7 (TCL.VerDec)**.** Given ciphertext qubits $\widetilde{\sigma_i}$ (with encrypted one-time pad keys $\widetilde{x[i]}, \widetilde{z[i]}$), a secret key (π, k, sk, pk), a circuit description c, and a classical computation log *log*, do the following:

1. Verify all classically authenticated messages (in *log*) using $\mathsf{MAC.Ver}_k$. If one of these verifications fails, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
2. Check whether all claimed gates in *log* match the structure of c. If not, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
3. Check the correctness of the classical log using $\mathsf{TCL.CheckLog}(log, pk, evk)$. If it is not correct, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
4. Check whether the claimed final QOTP keys in *log* match $\tilde{x}$ and $\tilde{z}$. If not, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
5. For all measurement gates of the circuit c, extract the encrypted measurement result $\tilde{b}$ and accept/reject flag $\widetilde{flag}$ from *log*. If $flag = \mathsf{rej}$, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$. Otherwise, append b to the output list.
6. For all unmeasured qubits $\widetilde{\sigma_i}$, do the following:
   (a) Decrypt the pad keys $x[i], z[i] \leftarrow \mathsf{HE.Dec}_{sk}(\widetilde{x[i]}, \widetilde{z[i]})$.
   (b) Run $\sigma_i \leftarrow \mathsf{TC.VerDec}_{(\pi, x[i], z[i])}(\widetilde{\sigma_i})$. If that rejects, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$. Otherwise, append σ_i to the output list.
7. Output the list of decrypted qubits and measurement outcome that are part of the output of c, together with the flag $|\mathsf{acc}\rangle\langle\mathsf{acc}|$.

Note that Step 7 is only reached if none of the previous steps trigger a reject. If even one of the checks fails (for example, one output qubit contains an error in the traps), then the entire procedure rejects.

5.6.1 Correctness, compactness, and privacy

If all classical computation was unencrypted, checking correctness of TCL could be done by inspecting the evaluation procedure for the different types of gates, and comparing them to the interactive protocol for quantum computing on authenticated data (Section 3.4.2). Since HE and the MAC authentication both satisfy correctness, the full scheme TCL is then also correct.

Compactness as in Definition 5.4.3 is also satisfied: verifying the computation log and checking all intermediate measurements (up until Step 4 in Protocol 5.6.7) is a completely classical procedure and runs in polynomial time in its input. The rest of TCL.VerDec only uses the secret key and the ciphertext $(\widetilde{\sigma}, \widetilde{x}, \widetilde{z})$ as input, not the log or the circuit description. Thus, we can separate TCL.VerDec into two algorithms Ver and Dec as described in Definition 5.4.3, by letting the second part (Dec, Steps 5 to 7) reject whenever the first part (Ver, Steps 1 to 4) does. It is worth noting that, because the key-update steps are performed homomorphically during the evaluation phase, skipping the classical verification step (i.e., only executing Dec) yields a QHE scheme without verification that satisfies compactness in the original sense of Definition 5.2.4. This is not the case for the scheme TC, where the classical computation is necessary for the correct decryption of the output state.

In terms of privacy, TCL satisfies q-IND-CPA (see Definition 5.2.9 for the public-key version: the private-key version is the same, except that the adversary $\mathcal{A}_1$ gets access to an encryption oracle instead of the encryption key). This type of security is shown in the same way as for CL (Section 5.3): by reduction to q-IND-CPA of the underlying classical scheme HE. Security of HE allows us to replace the classical information accompanying ciphertexts and magic states with the all-zero message. What is left is a collection of one-time-padded quantum states, where the one-time pad is freshly chosen at each new encryption. This contrasts with TC, where the one-time-pad key was contained in the secret key, posing a security risk when an encryption oracle was provided.

5.6.2 Secure verifiability

In this section, we will prove that TCL is IND-VER. By Lemma 5.4.8, it then follows that TCL is also verifiable in the semantic sense (SEM-VER). We will define a slight variation on the VER indistinguishability game, followed by several hybrid schemes (variations of the TCL scheme) that fit into this new game. We will argue that for any adversary, changing the game or scheme does not significantly affect the winning probability. After these hybrid steps, we will have reduced the adversary to an adversary for the somewhat homomorphic scheme TC, which we already know to be

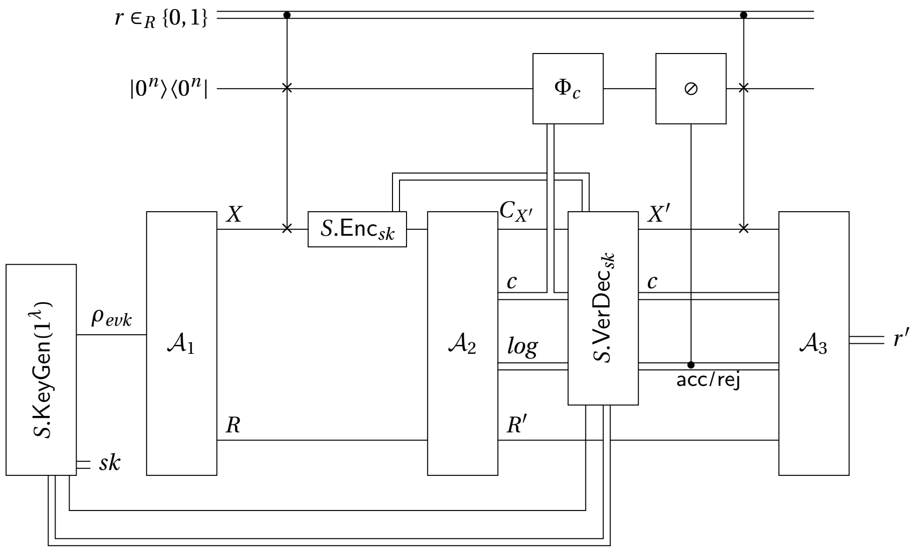


Figure 5.6: The hybrid indistinguishability game $\mathsf{Hyb}_{\mathcal{A},S}(\lambda)$, which is a slight variation on $\mathsf{VerGame}_{\mathcal{A},S}(\lambda)$ from Figure 5.3.

IND-VER. This will complete the argument that TCL is IND-VER.

Game 5.6.8 (Hybrid game $\mathsf{Hyb}_{\mathcal{A},S}(\lambda)$)**.** For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, a scheme S, and security parameter λ, $\mathsf{Hyb}_{\mathcal{A},S}(\lambda)$ is the game in Figure 5.6.

In comparison to Game 5.4.5 (see Figure 5.3), three new wires are added in this hybrid game: a classical wire from $S.\mathsf{Enc}$ to $S.\mathsf{VerDec}$, and a classical and quantum wire from $S.\mathsf{KeyGen}$ to $S.\mathsf{VerDec}$. We will later adjust TCL to use these wires to bypass the adversary[4]; TCL as defined above does not use them. Therefore, for any efficient adversary, $\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TCL}}(\lambda) = 1] = \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}}(\lambda) = 1]$.

Recall that all adversaries are QPTs, i.e., quantum polynomial-time uniform algorithms. Given two hybrid games H_1, H_2, and a QPT adversary $\mathcal{A}$, define the *advantage*

$$\mathsf{AdvHyb}_{H_1}^{H_2}(\mathcal{A},\lambda) := \left|\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{H}_1}(\lambda) = 1] - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{H}_2}(\lambda) = 1]\right|. \tag{5.36}$$

The goal will be to show that for each new hybrid, the advantage of the previous hybrid over the new one is negligible in λ.

[4]The quantum wire from $S.\mathsf{KeyGen}$ to $S.\mathsf{VerDec}$ will not be used in this chapter. It will be relevant for the proof of Theorem 6.4.13 in the next chapter, which is why we include it in the definition of Game 5.6.8.

Hybrid 1: Removing the classical MAC

In TCL, the initial keys to the QOTP can only become known to VerDec through the adversary. We thus use a MAC to make sure these keys cannot be altered. Without this authentication, the adversary could, e.g., homomorphically use $\widetilde{\pi}$ to flip only those bits in $\widetilde{x}$ that correspond to nontrap qubits, thus applying X to the plaintext. For the same reason, all classical information in the evaluation key must be authenticated.

In the first hybrid, we argue that the winning probability of a QPT $\mathcal{A}$ in $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}}(\lambda)$ is at most negligibly higher than in $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}'}(\lambda)$. TCL' is a modified version of TCL, where the initial keys are sent directly from KeyGen and Enc to VerDec (via the extra wires in Figure 5.6). More precisely, in $\mathsf{TCL}'.\mathsf{KeyGen}$ and $\mathsf{TCL}'.\mathsf{Enc}$, whenever $\mathsf{MAC.Sign}(\mathsf{HE.Enc}(x))$ or $\mathsf{MAC.Sign}(x)$ is called, the message x is *also* sent directly to $\mathsf{TCL}'.\mathsf{VerDec}$. Moreover, instead of decrypting the classically authenticated messages sent by the adversary, $\mathsf{TCL}'.\mathsf{VerDec}$ uses the information it received directly from $\mathsf{TCL}'.\mathsf{KeyGen}$ and $\mathsf{TCL}'.\mathsf{Enc}$. It still checks whether the computation log provided by the adversary contains these values at the appropriate locations and whether the MAC signature is correct. The following fact is then a direct consequence of the EUF-CMA property of MAC.

Lemma 5.6.9. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}_{\mathsf{TCL}}^{\mathsf{TCL}'}(\mathcal{A},\lambda) \leq \mathrm{negl}(\lambda)$.

Proof. Before VerDec is called, the games $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}}(\lambda)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}'}(\lambda)$ are completely identical: adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ receive the same inputs in both games, and therefore produce the same outputs. We thus only need to argue that $\mathsf{TCL}.\mathsf{VerDec}$ and $\mathsf{TCL}'.\mathsf{VerDec}$ have the same output distribution.

Given that the input from $\mathcal{A}_2$ to VerDec is identical in both games, the only way for the output of VerDec to differ, is if there is a signed message that passes $\mathsf{MAC.Ver}$, but decrypts to a different value (in TCL) than the value that is sent through the side channel (in TCL'). If such a value exists with nonnegligible probability, then we can define an adversary to the EUF-CMA security game[5] based on $\mathcal{A}_1, \mathcal{A}_2$, as follows.

The new adversary $\mathcal{A}'$ runs Game 5.6.8 (see Figure 5.6) until right before VerDec, using the adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ as subroutines. $\mathcal{A}'$ takes on the role of challenger, and computes the key generation, potential swap, and encryption herself. Any time the key generation or encryption requires $\mathcal{A}'$ to sign a message, she simply queries her EUF-CMA challenger for a signature. When $\mathcal{A}_2$ produces an output, the *log* contains several different signed messages (which would normally be checked by $\mathsf{TCL}.\mathsf{VerDec}$, and then decrypted for their values). $\mathcal{A}'$ randomly selects one of those signed messages, and outputs it.

[5]In the EUF-CMA ("existentially unforgeable under chosen message attacks") game, an adversary is allowed to request a polynomial number of signatures for messages that he chooses. He wins if he is able to produce a message m^*, together with a valid signature, such that m^* was not in the list of queried messages [KL14].

If $\mathcal{A}_2$ is able to produce *at least one* corrupted signed message with nonnegligible probability, then $\mathcal{A}'$ will pick that one (out of polynomially many options) with nonnegligible probability. In other words, if there is a nonnegligible difference between the winning probabilities in $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}}(\lambda)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}'}(\lambda)$, then $\mathcal{A}'$ can beat the EUF-CMA game with nonnegligible probability. □

Hybrid 2: Removing the computation log

In TCL and TCL', the adversary (homomorphically) keeps track of the keys to the QOTP and stores encryptions of all intermediate values in the computation log. Whenever VerDec needs to know the value of a key (for example to check a trap or to decrypt the final output state), the relevant entry in the computation log is decrypted.

In TCL', however, the plaintext initial values to the computation log are available to VerDec, as they are sent through the classical side channels. This means that whenever VerDec needs to know the value of a key, instead of decrypting an entry to the computation log, it can be computed by "shadowing" the computation log in the clear.

For example, suppose the log contains the encryptions $\widetilde{b_1}, \widetilde{b_2}$ of two initial bits, and specifies the homomorphic evaluation of XOR, resulting in $\widetilde{b}$ where $b = b_1 \oplus b_2$. If one knows the plaintext values b_1 and b_2, then one can compute $b_1 \oplus b_2$ directly, instead of decrypting the entry $\widetilde{b}$ from the computation log.

We now define a second hybrid, TCL'', which differs from TCL' exactly in this way: VerDec still verifies the authenticated parts of the log, checks whether the computation log matches the structure of c, and checks whether it is syntactically correct. However, instead of decrypting values from the log (as it does in Steps 5 and 6a), it computes those values from the plaintext initial values, by following the computation steps that are claimed in the log. By correctness of classical FHE, we then have the following.

Lemma 5.6.10. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TCL}''}_{\mathsf{TCL}'}(\mathcal{A}, \lambda) \leq \mathrm{negl}(\lambda)$.

Proof. Let s be the (plaintext) classical information that forms the input to the classical computations performed by the adversary: initial QOTP keys, secret keys and permutations, measurement results, et cetera. Let f be the function that the adversary computes on it in order to arrive at the final keys and logical measurement results. By correctness of HE, we have that

$$\Pr\left[\mathsf{HE.Dec}_{sk}\left(\mathsf{HE.Eval}^{f}_{evk}\left(\mathsf{HE.Enc}_{pk}(s)\right)\right) \neq f(s)\right] \leq \mathrm{negl}(\lambda). \tag{5.37}$$

Thus, the probability that $\mathsf{TCL}'.\mathsf{VerDec}$ and $\mathsf{TCL}''.\mathsf{VerDec}$ use different classical values (decrypting from the log vs. computing from the initial values) is negligible. Since this is the only place where the two schemes differ, the output of the two VerDec functions will be identical, except with negligible probability. Thus $\mathcal{A}$ will either win in both $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}'}(\lambda)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda)$, or lose in both, again except with negligible probability. □

Hybrid 3: Removing all classical FHE

In TCL'', the entire computation log is ignored, and all classical functionality has been redirected to VerDec, which receives the relevant information directly from KeyGen and Enc. The adversary still receives homomorphic encryptions of all relevant information, and is supposed to compute on them, because the log is checked for syntactical consistency. However, VerDec does not decrypt and use the resulting values. This fact allows us to link TCL'' to a final hybrid, TCL^*, where all classical information is replaced with zeros before encrypting. That is, in $\mathsf{TCL}^*.\mathsf{Enc}$, signed encryptions of zeros are appended (instead of signed encryptions of x, z), and in $\mathsf{TCL}^*.\mathsf{KeyGen}$, the evaluation key contains an encryption of zero instead of an encryption of π.

Lemma 5.6.11. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TCL}^*}_{\mathsf{TCL}''}(\mathcal{A},\lambda) \leq \mathrm{negl}(\lambda)$.

Proof. We prove the statement by reduction to q-IND-CPA security of the classical scheme HE. Technically, one would have to do this in two steps, via yet another hybrid: first by replacing the encryptions of x, z keys inside Enc with encryptions of zeros, then the encryptions of magic states and π inside KeyGen. In this proof, we focus on the first step, by assuming that TCL^* still encrypts normally during KeyGen. The second step would work in very much the same way.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be an adversary for the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda)$. Note that we may interpret $\mathcal{A}$ as an adversary for $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda)$ as well, since the games are syntactically identical. We will define an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the *public*-key game $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$. This adversary will simulate the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda)$ or $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda)$: which game is simulated is unknown to $\mathcal{A}'$, and depends on a random choice $s \in \{0,1\}$ of the challenger for $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$.

We define the adversary $\mathcal{A}'$ in two parts (see also Figure 5.7):

$\mathcal{A}'_1$ receives a classical public key *pk* and evaluation key *evk* from her challenger. She runs $\mathsf{TCL}.\mathsf{KeyGen}$, using *pk*, *evk* instead of the output of $\mathsf{HE}.\mathsf{KeyGen}$. Note that *sk* is not actually used during KeyGen: for the encryptions, the classical secret key is not relevant. We can safely set it to a dummy string for now. $\mathcal{A}'_1$ feeds the generated evaluation key to $\mathcal{A}_1$, and sends the generated secret key (π, k, sk, pk) to $\mathcal{A}'_2$ via their side channel. $\mathcal{A}_1$ outputs a challenge plaintext.

$\mathcal{A}'_1$ flips a bit $r \in_R \{0,1\}$. If $r = 1$, she swaps out the challenge plaintext for an all-zero state (see Figure 5.6). The remaining state on the message wire is encrypted using $\mathsf{TCL}.\mathsf{Enc}$, except that $\mathcal{A}'_1$ does not run $\mathsf{HE}.\mathsf{Enc}_{pk}$ herself. Instead, she sends the one-time-pad keys x, z to her own challenger for encryption. That

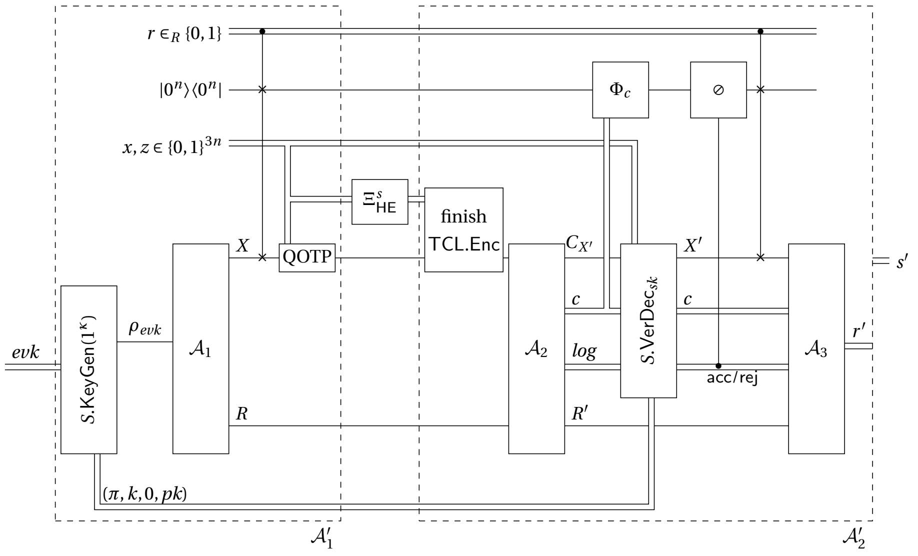


Figure 5.7: The adversary $\mathcal{A}'$ for the game $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$, defined in terms of $\mathcal{A}$. The first part ($\mathcal{A}'_1$) receives an evaluation key *evk* from $\mathsf{HE.KeyGen}(1^\lambda)$ (not shown). The encryption of the challenge plaintext in the X register is broken up into three parts: a quantum one-time pad ("QOTP", part of $\mathcal{A}'_1$), a classical encryption by the challenger ($\Xi^s_{\mathsf{HE}} = \Xi^{\mathsf{cpa-mult},s}_{\mathsf{HE}}$, see Game 5.2.8), and the part that applies the MAC to the classical encryptions and concatenates it with the quantum state ("finish $\mathsf{TCL.Enc}$", part of $\mathcal{A}'_2$). The adversary $\mathcal{A}'_2$ outputs $s = 1$ whenever $r' = r$.

challenger selects a secret bit $s \in \{0,1\}$, and either encrypts the actual values x, z (if $s = 1$), or the all-zero message (if $s = 0$). $\mathcal{A}'_1$ sends the plaintext keys x, z to $\mathcal{A}'_2$.

$\mathcal{A}'_2$ receives the classically encrypted one-time-pad keys (or zeros) from the challenger, and uses them to finish the encryption of the challenge submitted by $\mathcal{A}_1$. The result is passed on to $\mathcal{A}_2$, who produces an output ciphertext, circuit, and log. $\mathcal{A}'_2$ checks these using $\mathsf{TCL}^*.\mathsf{VerDec}$ (or $\mathsf{TCL}''.\mathsf{VerDec}$, which is identical). Here, we note that this VerDec function does not use the classical secret key *sk*, so it is fine that $\mathcal{A}'_2$ does not actually know it.

$\mathcal{A}'_2$ continues the verification game, applying the circuit to the swapped-out register, and potentially deleting it, before swapping it back in if $r = 1$. The result is sent to $\mathcal{A}_3$, who outputs a guess r'. If the guess is correct ($r' = r$), then $\mathcal{A}'_2$ outputs her own guess $s' := 1$. If not, she outputs $s' := 0$.

By the q-IND-CPA security of HE, we know that this newly defined adversary cannot have a high success probability of guessing s' correctly. That is,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda). \tag{5.38}$$

We analyze the probability $\Pr[\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda) = 1]$ by breaking it up into the two possible values for s, which both occur with equal probability. If $s = 1$, the simulated game is $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda)$. $\mathcal{A}'$ will correctly output $s' = 1$ if and only if $\mathcal{A}$ correctly guesses the value r. On the other hand, if $s = 0$, the simulated game is $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda)$: in that case, $\mathcal{A}'$ will output the correct answer $s' = 0$ if and only if $\mathcal{A}$ *loses*, i.e., incorrectly guesses r.

Thus,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda) = 1] \tag{5.39}$$
$$= \Pr[s = 1] \cdot \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda) = 1] + \Pr[s = 0] \cdot \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 0] \tag{5.40}$$
$$= \frac{1}{2} \cdot \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}''}(\lambda) = 1] + \frac{1}{2}\left(1 - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1]\right). \tag{5.41}$$

Combining the above derivation with Equation (5.38), the statement follows.

We briefly comment on the next step of this proof: removing the classical encryptions inside KeyGen. It is similar to the proof above: one can define another adversary for the q-IND-CPA indistinguishability experiment similarly to $\mathcal{A}'$, except that she submits her challenge plaintexts during KeyGen instead of during Enc. The challenger then decides (based on the choice of s) whether to encrypt the classical information in the evaluation key, or to replace it with zeros. During encryption, the newly defined adversary will always encrypt zeros. With the same argument as above (using q-IND-CPA security of HE), it can be argued that no adversary can perform significantly better when it is given "meaningful" encryptions in the evaluation key than when it is given encryptions of zeros. □

Proof of main theorem

Considering TCL^* in more detail, we can see that it is actually very similar to TC. This similarity allows us to prove the following lemma, which is the last ingredient for the proof of verifiability of TCL.

Lemma 5.6.12. *For any QPT* $\mathcal{A}$, $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1] \leq \frac{1}{2} + \mathrm{negl}(\lambda)$.

Proof. To see the similarity with TC, consider the four algorithms of TCL^*.

In $\mathsf{TCL}^*.\mathsf{KeyGen}$, a permutation π is sampled, and magic states for P and H are generated. For all generated quantum states, random keys for QOTPs are sampled, and the states are encrypted using $\mathsf{TC.Enc}$ with these keys as secret keys. No classical FHE is present anymore. Thus, $\mathsf{TTP}^*.\mathsf{KeyGen}$ can be viewed as $\mathsf{TC.KeyGen}$, followed by $\mathsf{TC.Enc}$ on the magic states.

$\mathsf{TCL}^*.\mathsf{Enc}$ is identical to $\mathsf{TC.Enc}$, only the keys to the quantum one-time pad are sampled on the fly and sent to $\mathsf{TCL}^*.\mathsf{VerDec}$ via a classical side-channel, whereas $\mathsf{TC.VerDec}$ receives them as part of the secret key. Since the keys are used exactly once and not used anywhere else besides in Enc and VerDec, this difference does not affect the outcome of the game.

$\mathsf{TCL}^*.\mathsf{Eval}$ only needs to evaluate "simple" circuits consisting of CNOT, classically controlled Paulis, computational-basis measurements and Hadamard-basis measurements. For the execution of any other gate, it suffices to apply a "simple" circuit to the encrypted data and encrypted magic states.

$\mathsf{TCL}^*.\mathsf{VerDec}$ does two things: (1) it syntactically checks the provided computation log, and (2) it runs $\mathsf{TC.VerDec}$ to verify that the evaluation procedure correctly applied the circuit of CNOTs and measurements.

An execution of $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda)$ for any $\mathcal{A}$ corresponds to the two-round VER indistinguishability game (see Definition 5.5.4) for TC as follows. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be a polynomial-time adversary for the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda)$. Define an additional QPT $\mathcal{A}_0$ that produces magic states into the register X_1. The above analysis shows that the adversary $\mathcal{A}' = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ can be viewed as an adversary for the VER-2 indistinguishability game $\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\lambda)$ and wins whenever $\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1$. The other direction does not hold: $\mathcal{A}$ loses the hybrid indistinguishability game if $\mathsf{TCL}^*.\mathsf{VerDec}$ rejects check (1), but accepts check (2) (see above). In this case, $\mathcal{A}'$ would still win the VER-2 indistinguishability game. Hence,

$$\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1] \leq \Pr[\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\lambda) = 1]. \tag{5.42}$$

Lemma 5.5.6 yields $\Pr[\mathsf{VerGame}^2_{\mathcal{A}',\mathsf{TC}}(\lambda) = 1] \leq 1/2 + \mathrm{negl}(\lambda)$, and the result follows. □

Now we finally have all the ingredients we need to prove our main theorem:

Theorem 5.6.13. *The vQHE scheme* TCL *satisfies SEM-VER.*

Proof. From Lemmas 5.6.9 to 5.6.11, we can conclude that for any polynomial-time quantum adversary $\mathcal{A}$,

$$\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TCL}}(\lambda) = 1] \; - \; \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1] \;\leqslant\; \mathrm{negl}(\lambda), \tag{5.43}$$

since the sum of a constant number of negligible terms is itself negligible.

By Lemma 5.6.12, which reduces verifiability of TCL^* to (two-round) verifiability of TC, we have that $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TCL}^*}(\lambda) = 1] \leqslant 1/2 + \mathrm{negl}(\lambda)$.

It follows that $\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TCL}}(\lambda) = 1] \leqslant 1/2 + \mathrm{negl}(\lambda)$, i.e., that TCL is IND-VER. By Lemma 5.4.8, TCL is also SEM-VER. □

5.7 Conclusion

In this chapter, we laid the groundwork for (verifiable) quantum homomorphic encryption by presenting schemes that can evaluate any (polynomial-size) Clifford circuit on an encrypted input quantum state. In the nonverifiable setting, we discussed the existing scheme CL, and showed that it can be adapted to a circuit-private version simply by having the evaluator apply an extra Pauli at the end of the computation.

For the verifiable setting, we set up a new framework by defining a semantic version of secure verifiability (SEM-VER), in addition to a game-based one (IND-VER). The two definitions are equivalent, but it is desirable to have them both. On the one hand, the semantic definition better reflects the functionality that we are trying to achieve, and allows replacing any secure vQHE scheme with the ideal functionality when composing it with other functionalities. On the other hand, the game-based definition is easier to deal with in reduction proofs, of which we have seen many in this chapter.

We also discussed compactness in the verifiable setting: while compactness usually requires that the running time of the decryption function is independent of the evaluated circuit, this requirement seems too harsh in a setting where the decryption function is also supposed to *verify* the circuit: in particular, it should probably at least be able to read out the circuit description. We solved this apparent contradiction by allowing the decryption function to run a *classical* verification procedure that is allowed to depend on the circuit size, but should not help in decrypting the output state. It may be possible to further reduce the complexity of this verification function by only checking a small number of random positions in the computation log, similarly to a probabilistically checkable proof [AS98].

In this chapter we also presented a specific scheme, TCL, instantiating our vQHE definition for Clifford circuits. It combines ideas from the nonverifiable scheme CL with the trap code for authenticating quantum data. The security of TCL relies on the security of a building-block scheme, TC, which is information-theoretically secure but not compact.

The obvious next step is to expand the schemes CL and TCL to be able to deal with arbitrary efficient quantum circuits. Doing so will be the focus of the next chapter.

6 Quantum homomorphic encryption for general circuits

Chapter contents

6.1 Introduction

In Chapter 5, we saw how to encrypt quantum states in such a way that Clifford operations can be performed homomorphically on the ciphertexts, both without and with verification. The obvious open question is whether it is possible to construct quantum *fully* homomorphic encryption, that allows the evaluation (and verification) of an arbitrary polynomial-size quantum circuit. This question boils down to defining an evaluation procedure for the non-Clifford gate T. If, in the spirit of the Clifford scheme CL (described in Section 5.3), the evaluator simply applies the gate T to the input ciphertext, the result is

$$\mathsf{T}\mathsf{X}^a\mathsf{Z}^b|\psi\rangle = \mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}|\psi\rangle. \tag{6.1}$$

Although this state contains the desired logical state $\mathsf{T}|\psi\rangle$, it is not a quantum-one-time-pad encryption of it: depending on the value of the encryption key a, the output state may contain an unwanted phase P. This unwanted phase is notoriously hard to get rid of: the evaluator does not (and should not!) know the value a, but rather only holds an encryption $\widetilde{a}$, which it produced using classical FHE. Therefore, she cannot remove the P^a. However, she also cannot continue with the computation (not even to evaluate Clifford circuits) until the possible phase is removed, because it does not commute nicely with most gates.

Broadbent and Jeffery [BJ15] proposed two different approaches for extending the Clifford scheme CL, accomplishing homomorphic encryption for circuits with a limited number of T gates. These two schemes use different methods to remove the conditional phase P^a, which we briefly discuss here.

In the scheme EPR, some entanglement is accumulated in a special register during every evaluation of a T gate, and stored there until it can be resolved in the decryption phase. Essentially, the decrypting party can remove P^a "after the fact" by manipulating the entangled states. Because the decryption procedure handles the corrections, the scheme is not compact: the complexity of decryption scales quadratically with the number of T gates in the evaluated circuit.

The scheme AUX also extends CL, but handles T gates in a different manner. The evaluator is supplied with auxiliary quantum states, stored in the evaluation key. These states allow him to evaluate T gates and immediately remove any phase error that may have occurred. In this way, the decryption procedure remains very efficient, and the scheme is compact. Unfortunately, the required auxiliary states grow doubly exponentially in size with respect to the T depth of the circuit, rendering AUX useful only for circuits with constant T depth.

6.1.1 Contributions

We show how to extend CL (and later TCL) so that it allows evaluation of a polynomial number of T gates. In both schemes presented in this chapter, the client constructs input-independent quantum states, which we call *gadgets*. The gadgets are stored in the evaluation key, and aid in the evaluation of a T gate. The complexity of the key generation scales with the circuit size, but the decryption function is independent of it: thus, the schemes are compact. For an overview of the homomorphic-encryption schemes discussed in this dissertation, see Figure A on page 289.

T-gate gadget (Section 6.2). Every T gate requires exactly one evaluation gadget to correct the potential phase error P^a. The size of a gadget depends only on (a certain form of) the space complexity of the decryption function of the underlying classical FHE scheme. This relation turns out to be very convenient, as classical FHE schemes are often optimized with respect to the complexity of the decryption operation (in order to make them bootstrappable).

The quantum part of the gadget consists of EPR pairs, which are prepared in a way that depends on the secret key of the classical FHE scheme. The evaluator can teleport the encrypted qubit "through the gadget" [GC99; BFK09; BJ15; BY20] in a way that depends on $\tilde{a}$, in order to remove the unwanted phase. Some classical information is provided with the gadget that allows the evaluator to homomorphically update the encryption keys after the teleportation steps.

On a high level, the use of an evaluation gadget corresponds to an *instantaneous nonlocal quantum computation*[1] where one party holds the secret key of the classical FHE scheme, and the other party holds the input qubit and a classical encryption of the key to the quantum one-time pad. Together, this information determines whether an inverse phase gate $\mathsf{P}^\dagger$ needs to be performed on the qubit or not. Speelman [Spe16] shows how to perform such computations with a bounded amount of entanglement. Speelman's techniques are crucial to our construction and are the reason why the *garden-hose complexity* [BFSS13] of the decryption procedure of the classical FHE is related to the size of our gadgets.

Quantum fully homomorphic encryption (Section 6.3). We define the first quantum fully-homomorphic-encryption scheme, which we call TP (for teleportation). The scheme is secure against chosen-plaintext attacks by quantum adversaries (q-IND-CPA). Furthermore, TP does not depend on a specific classical FHE scheme, hence any advances in classical FHE can directly improve our scheme. Our requirements for the classical FHE scheme are quite modest: we only require the classical scheme to have a space-efficient decryption procedure and to be secure against quantum adversaries. In particular, no circular-security assumption is required. Since we supply at most a polynomial number of evaluation gadgets, our scheme TP is leveled homomorphic by construction, and we can simply switch to a new classical key after every evaluation gadget. In fact, the Clifford gates in the quantum evaluation circuit

[1]This term is not related to the term "instantaneous quantum computation" [SB08], but refers to a specific form of nonlocal quantum computation where all parties have to act simultaneously.

only require additive operations from the classical homomorphic scheme, while each T gate needs a fixed (polynomial) number of multiplications. Hence, we do not actually require fully homomorphic classical encryption, but leveled fully homomorphic schemes suffice.

Our scheme TP is related to AUX in that extra resources for removing errors are stored in the evaluation key. In sharp contrast to AUX, the size of the evaluation key in TP only grows linearly in the number of T gates in the circuit (and polynomially in the security parameter), allowing our scheme to be leveled fully homomorphic. Any circuit containing polynomially many T gates can be efficiently evaluated.

Subsequently to the results presented in this chapter, schemes have been designed for quantum fully homomorphic encryption that are less demanding for the client. We discuss these follow-up works in Section 6.7.1.

Quantum fully homomorphic encryption with verification (Section 6.4). We construct another scheme, called TTP, which extends the verifiable scheme TCL to a quantum fully-homomorphic-encryption scheme with verification (vQFHE). The server can certify the correct usage of the gadgets through its classical computation log. Just like TCL, the verification of this computation log is completely classical. Circuits with a classical output can be verified completely classically.

The scheme TTP admits *verified* evaluation of arbitrary polynomial-size quantum circuits. The scheme combines techniques and ideas from CL, TCL, and TTP. The main technical ingredients are (1) classical FHE with decryption in LOGSPACE [BV14], (2) the trap code for computing on authenticated quantum data, and (iii.) the T-gate gadgets that were originally designed for TP.

Applying a T gate in TTP requires an encoded magic state (just like P and H did in TCL), plus an encoded gadget to correct a potential phase error P. The key generator supplies both of these resources via the evaluation key.

Application: quantum one-time programs (Section 6.5). A one-time program (or OTP) is a device which implements a circuit, but self-destructs after the first use. OTPs are impossible without hardware assumptions, even with quantum states, but OTPs that implement quantum circuits (qOTP) can be built from classical OTPs (cOTPs) based on hardware assumptions [BGS13]. As a first application of verifiable QFHE, we give another simple construction of qOTPs. Our construction is weaker, since it requires a computational assumption. On the other hand, it is conceptually very simple and serves to demonstrates the power of verification. In our construction, the qOTP for a quantum circuit c is simply a (vQFHE) encryption of c together with a cOTP for verifying the universal circuit. To use the resulting qOTP, the user attaches their desired input, homomorphically evaluates the universal circuit, and then plugs their computation log into the cOTP to retrieve the final decryption keys.

Practical considerations (Section 6.6). As a concrete example, we instantiate our scheme with the classical FHE scheme by Brakerski and Vaikuntanathan [BV14], which has decryption in $\mathsf{NC}^1 \subseteq \mathsf{LOGSPACE}$, and is believed to be quantum secure. This results in gadgets that consist of a number of qubits which is polynomial in the security parameter, for a polynomial that is potentially smaller than the polynomial given by our generic construction.

Furthermore, we describe how the key generator for TP can generate the T-gate gadgets using a limited set of available quantum operations, provided it has an (untrusted) source or EPR pairs.

6.2 The T-gate gadget

Recall that when a T gate is applied to a state $\mathsf{X}^a\mathsf{Z}^b\left|\psi\right\rangle$, an unwanted phase error may occur, since $\mathsf{T}\mathsf{X}^a\mathsf{Z}^b = \mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}$. The evaluating party only knows the encrypted value $\widetilde{a}$, not a itself, and therefore is not easily able to remove the error P^a before continuing the computation. In this section, we show how the client (generating the key) can create a *gadget state* $\Gamma(sk)$ that depends on the secret key sk, but not on a. During the computation, the evaluator can use the quantum state $\Gamma(sk)$ in a way dictated by $\widetilde{a}$ to get rid of the phase error.

The construction of our T-gate gadget depends on results in the garden-hose model and instantaneous nonlocal quantum computation, which we explain in Sections 6.2.1 and 6.2.2. In Section 6.2.3, we specify the structure of the gadget. After usage, the error P^a is removed, but the one-time pad may have changed. In Section 6.2.4, we discuss how the evaluator can update the key homomorphically.

6.2.1 Garden-hose complexity

The *garden-hose model* is a model of communication complexity introduced by Buhrman, Fehr, Schaffner and Speelman to study position-based quantum cryptography [BFSS13]. In brief, the garden-hose model involves two parties: Alice, with input x, and Bob, with input y. Alice and Bob want to jointly compute a function $f(x,y)$ without communicating. They do share a number of one-to-one pipes between them, which they are allowed to connect on their respective sides. Alice also has a water tap, which she connects to one of the pipes. The way that Alice and Bob connect their pipes may depend on their local inputs x and y. Whenever $f(x,y) = 0$, the water should exit at an open pipe on Alice's side, and whenever $f(x,y) = 1$, the water should exit on Bob's side. For an example of a garden-hose computation for the XOR function, see Figure 6.1.

The *garden-hose complexity* $GH(f)$ of a function f is the minimal number of shared pipes needed for Alice and Bob to correctly compute the function for all inputs

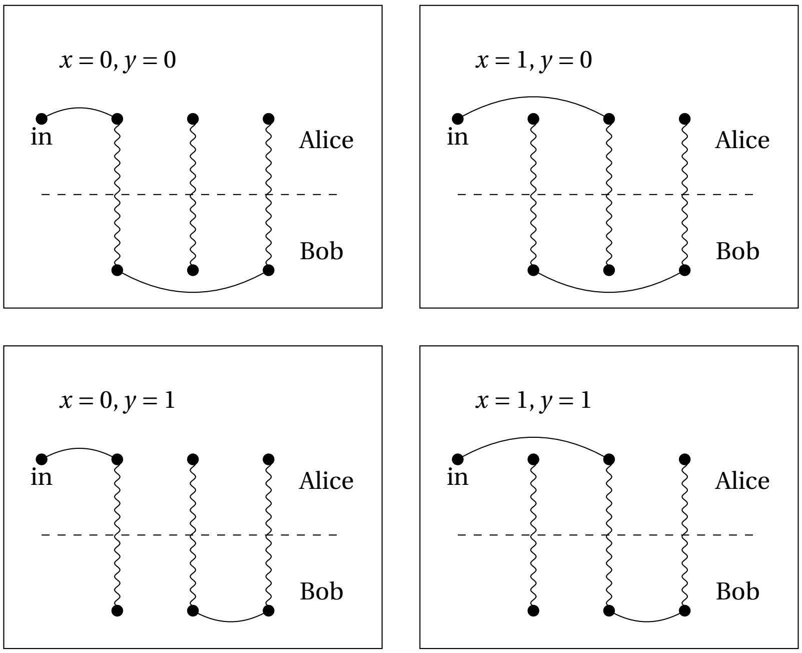


Figure 6.1: The garden-hose protocol for the function $f(x, y) = x \oplus y$, for all four possible inputs $(x, y) \in \{0, 1\}^2$. The snaky lines represent the pipes, and the smooth curved lines represent the connections that Alice (top) and Bob (bottom) make between their pipes. The node marked "in" is Alice's tap, which she always connects to one of the pipes. Note that Alice's strategy does not depend on y, and Bob's strategy does not depend on x. The water ends up on Alice's side whenever $x \oplus y = 0$. The depicted protocol demonstrates that $GH(\oplus) \leqslant 3$.

x and y. For some specific functions, such as equality, majority, parity, and bitwise inner product, concrete garden-hose computation protocols are known [Spe11; BFSS13; CSWX14; Mar14; KP14], giving an upper bound on their garden-hose complexity. The following theorem provides us with a general way of transforming space-efficient algorithms into garden-hose protocols:

Theorem 6.2.1 ([BFSS13, Theorem 2.12])**.** *If $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is* LOGSPACE *computable, then $GH(f)$ is polynomial in n.*

Since the garden-hose complexity is defined in a nonuniform way, the strategies of the players are not necessarily easily computable. However, by inspection of the original proof of Theorem 6.2.1, we see that the players effectively have to list all configurations for the Turing machine for f, and connect them according to the machine's transition function. For a LOGSPACE function on a fixed input size, a player therefore only has to perform a polynomial-time computation to determine the strategy for a specific input.

6.2.2 Instantaneous nonlocal quantum computation

Speelman combined the garden-hose model with techniques from secure delegated computation [Bro16] to construct a new protocol for instantaneous nonlocal quantum computation [Spe16]. Here, Alice and Bob want to perform a unitary operation on a joint quantum state, using only pre-shared entanglement and a single round of classical communication. This new protocol breaks a wider class of schemes for position-based quantum cryptography, but one of its subprotocols also serves as a building block for the T-gate gadgets described in this chapter.

In the subtask, Alice and Bob want to apply the Clifford gate $\mathsf{P}^\dagger$ to a state held by Alice, *conditioned* on the value of $f(x,y)$. Speelman showed that this is possible (up to a Pauli error) using a number of EPR pairs that is proportional to the garden-hose complexity of f:

Lemma 6.2.2 ([Spe16, Lemma 3, part 1])**.** *Assume Alice has a single qubit with state $\mathsf{P}^{f(x,y)}|\psi\rangle$, for binary strings $x, y \in \{0,1\}^n$, where Alice knows x and Bob knows y. Then there exists an instantaneous protocol without any communication which uses $2GH(f)$ pre-shared EPR pairs after which a known qubit of Alice is in the state $\mathsf{X}^{g(\hat{x},\hat{y})}\mathsf{Y}^{h(\hat{x},\hat{y})}|\psi\rangle$. Here $\hat{x}$ depends only on x and the measurement outcomes of Alice, and $\hat{y}$ depends on y and the measurement outcomes of Bob.*

We explain the intuition behind the construction in Lemma 6.2.2: Alice and Bob essentially execute the garden-hose protocol for f, but with two extra ingredients.

The first ingredient is a close correspondence between garden-hose protocols (when Alice and Bob share pipes) and teleporting qubits back and forth (when they share EPR pairs). The pipes correspond to EPR pairs, and the connections to Bell measurements [BFSS13]. Say that Bob's garden-hose protocol instructs him to connect

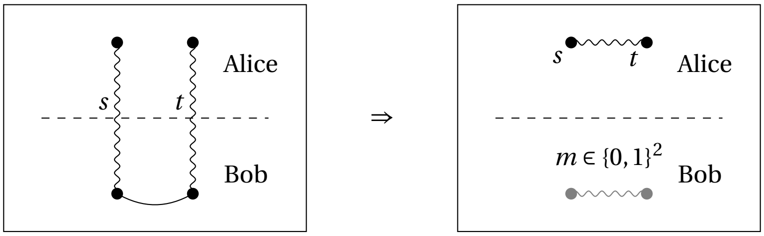


Figure 6.2: An entanglement-swapping measurement. The two snaky lines on the left-hand side now represent EPR pairs s and t, which are in the joint state $|\Phi^+\rangle^{\otimes 2}$. Bob measures one qubit of s and one qubit of t in the Bell basis, obtaining measurement a two-bit measurement outcome m. He discards the measured qubits (depicted in gray on the right-hand side). The two remaining qubits on Alice's side are in the state $\mathsf{X}^{m[1]}\mathsf{Z}^{m[2]}|\Phi^+\rangle$.

pipes s and t. Instead, he measures his halves of the EPR pairs s and t in the Bell basis. As a result, Alice's two halves of the pairs s and t are now fully entangled: Bob has effectively teleported his half of the pair s to Alice, through the pair t. This teleportation, sometimes called *entanglement swapping*, is analogous to Bob connecting the pipes s and t, which essentially creates a direct water connection between Alice's two ends of those pipes. See Figure 6.2. Which of the four Bell states is formed, depends on the two-bit outcome of Bob's measurement, describing the X and Z teleportation corrections.

Alice and Bob can execute a full garden-hose protocol by measuring pairs of EPR halves at all locations where they have connected pipes. The input qubit $\mathsf{P}^{f(x,y)}|\psi\rangle$ plays the role of Alice's tap: she teleports it through the EPR half corresponding to the pipe that should be connected to the tap. As a result, the qubit ends up at one of the unmeasured locations on Alice's side (if $f(x,y) = 0$) or Bob's side (if $f(x,y) = 1$). Bob applies an inverse phase gate $\mathsf{P}^\dagger$ to all of his unmeasured locations, so that the qubit is in the state $|\psi\rangle$ (up to a Pauli error), regardless of whose side it ends up in. The uncorrected Pauli on the qubit is a function of all the teleportation measurement results, and depends on the commutation relation between P and the Pauli group. The problem is, of course, that Alice and Bob do not know who holds the qubit, and at which location.

The second ingredient is a trick inspired by results on the garden-hose model by Klauck and Podder [KP14]. The trick ensures that the qubit ends up at a known location on Alice's side, at the cost of needing $2GH(f)$ EPR pairs rather than $GH(f)$. Using the second set of $GH(f)$ EPR pairs (or pipes), Alice and Bob run the *reverse* of the garden-hose protocol (see Figure 6.3): every EPR half on which no measurement is performed is connected through measurement with the EPR half at the same position in a second copy of the protocol. Only the EPR pair that Alice is supposed to connect to the tap remains unmeasured: this is where the qubit ends up, after following its

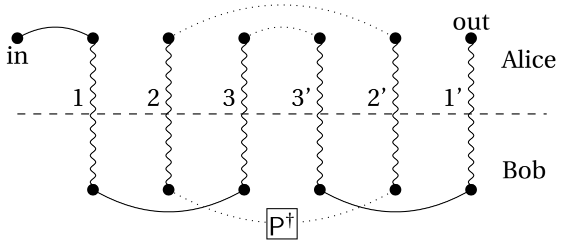


Figure 6.3: Two copies of the garden-hose protocol for the XOR function with inputs $x = y = 0$. The tap for the forward protocol is labeled "in", and the qubit that should be connected to a tap for the backward protocol (displayed in mirror here) is labeled "out". The dotted lines represent the Bell measurements that connect the two protocols: all unmeasured EPR pairs i in the forward protocol are connected to the corresponding pair i' in the backward protocol via Bell measurement. Additionally, Bob adds a correction $\mathsf{P}^\dagger$ on the unmeasured qubits in the forward protocol before measuring. In this case, since $x \oplus y = 0$, the qubit does not pass through the $\mathsf{P}^\dagger$ gate.

path through the garden-hose protocol, a possible P correction, and the same path backwards. The output qubit is in the state $\mathsf{X}^{g(\hat{x},\hat{y})}\mathsf{Y}^{h(\hat{x},\hat{y})}\left|\psi\right\rangle$, where the Pauli error $\mathsf{X}^{g(\hat{x},\hat{y})}\mathsf{Y}^{h(\hat{x},\hat{y})}$ now also depends on the outcomes of the measurements on the second set of EPR pairs.

6.2.3 Gadget construction

We are ready to specify the structure of the gadget for the T gate. In the language of Sections 6.2.1 and 6.2.2, Bob corresponds to the party that creates the gadget, so he has knowledge of the secret key sk. Alice corresponds to the party using the gadget, so she has access to $\widetilde{a}$ and a state $\mathsf{P}^a\left|\psi\right\rangle$, where $\mathsf{HE.Dec}_{sk}(\widetilde{a}) = a$. Note that for simplicity of notation, we have absorbed the Pauli $\mathsf{X}^a\mathsf{Z}^b$ into the state $\left|\psi\right\rangle$.

The function f that Alice and Bob will execute together is $\mathsf{HE.Dec}$: they each know part of the input $(sk, \widetilde{a})$, and Lemma 6.2.2 guarantees the existence of an instantaneous protocol so that at the end of the protocol, Alice holds the state $\mathsf{X}^{a'}\mathsf{Z}^{b'}\left|\psi\right\rangle$, where a' and b' are functions of sk, $\widetilde{a}$, and Alice and Bob's measurement outcomes.

A crucial observation is that since the protocol is instantaneous (i.e., there is no communication between Alice and Bob during the protocol), the order of the actions does not matter: Bob can perform his part of the protocol (Bell measurements and application of $\mathsf{P}^\dagger$ gates) before Alice even starts her first measurement. Using this observation, Bob (the key generator) can generate a gadget state as follows: he prepares $2GH(\mathsf{HE.Dec})$ EPR pairs, and executes his part of the instantaneous-computation protocol on the halves of each pair. Note that all halves on Bob's side are measured. The remaining halves, which correspond to Alice's side of the protocol, constitute

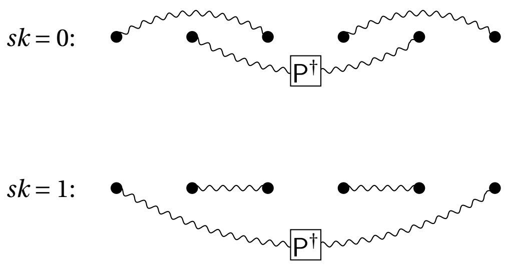


Figure 6.4: The two possible gadgets for a decryption function $\mathsf{Dec}(sk, c) := sk \oplus c$. (For more realistic decryption functions, there will be more than two possible keys, and therefore more than two possible gadget structures.) Depending on the value of sk, the key generator measures according to Bob's garden-hose protocol for $y = 0$ (top) or $y = 1$ (bottom) in Figure 6.1. The snaky lines represent random Bell pairs (the type of Bell pairs is determined by Bob's measurement outcomes). In this example, $g(0) = (\{(1,3),(2,5),(4,6)\}, 010, 0)$, and $g(1) = (\{(1,6),(2,3),(4,5)\}, 100, 1)$. The top case ($sk = 0$) corresponds to Bob's measurements in Figure 6.3.

the gadget state. See Figure 6.4. During the computation, Alice (the evaluator) can execute her half of the protocol to remove the phase gate from a qubit she holds.

More specifically, let $m = GH(\mathsf{HE.Dec})$, i.e., $2m$ is the number of EPR pairs required for the instantaneous protocol. Let $\{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}$ be the set of disjoint pairs in $[2m]$ that specify which connecting measurements Bob should make according to his part of the protocol. Let $p \in \{0,1\}^m$ be a string that specifies which of the first m pairs (i.e., the pairs involved in the "forward" garden-hose protocol) should receive a $\mathsf{P}^\dagger$ correction. Then, we can define

$$g(sk) := \big(\{(s_1, t_1), (s_2, t_2), \ldots, (s_m, t_m)\}, p, sk\big) \tag{6.2}$$

to be the tuple consisting of the connection instructions, phase-gate positions, and secret key. The tuple $g(sk)$ is the classical information that determines the structure of the gadget, as a function of the secret key sk. The length of $g(sk)$ is not dependent on the value of the secret key, but only on the garden-hose complexity of $\mathsf{HE.Dec}$ and the security parameter λ.

After Bob (the key generator) prepares and measures the EPR pairs according to the information in $g(sk)$, the resulting quantum state is of the form

$$\gamma_{x,z}\big(g(sk)\big) = \left(\mathsf{X}^x\mathsf{Z}^z\left(\mathsf{P}^\dagger\right)^p \otimes \mathsf{I}^{\otimes m}\right)\left(\bigotimes_i |\Phi^+\rangle\langle\Phi^+|_{s_i t_i}\right)\left(\mathsf{P}^p\mathsf{Z}^z\mathsf{X}^x \otimes \mathsf{I}^{\otimes m}\right), \tag{6.3}$$

where the single-qubit gates are applied to the s_i qubits, i.e., the first qubit of each entangled pair. The values $x, z \in \{0,1\}^m$ depend on the measurement outcomes of Bob, and are uniformly random.

This quantum state is a collection of maximally entangled pairs, some with an extra inverse phase gate applied. No matter the choice of gadget structure, averaging over all possible x, z gives the completely mixed state on $2m$ qubits:

$$\frac{1}{2^{2m}} \sum_{x,z \in \{0,1\}^m} \gamma_{x,z}\big(g(sk)\big) = \frac{\mathbb{I}_{2^{2m}}}{2^{2m}}. \tag{6.4}$$

This property will be important in the security proof: intuitively, it shows that these gadgets do not reveal any information about sk whenever x and z are not revealed to Alice.

The entire gadget is given by the state $\gamma_{x,z}\left(g(sk)\right)$, plus the classical information x, z, and $g(sk)$. All classical information is homomorphically encrypted under a public key pk':

$$\begin{aligned}\Gamma_{pk'}(sk) := \Big|\mathsf{HE.Enc}_{pk'}\left(g(sk)\right)\Big\rangle\Big\langle \mathsf{HE.Enc}_{pk'}\left(g(sk)\right)\Big| \otimes \\ \frac{1}{2^{2m}} \sum_{x,z \in \{0,1\}^m} \Big|\mathsf{HE.Enc}_{pk'}(x,z)\Big\rangle\Big\langle \mathsf{HE.Enc}_{pk'}(x,z)\Big| \otimes \gamma_{x,z}\left(g(sk)\right).\end{aligned} \tag{6.5}$$

Since the gadget depends on the secret key sk, simply encrypting this information using the public key pk corresponding to sk would not be secure, unless we assume that HE.Dec is circularly secure. In order to avoid the requirement of circular security, we will always use a fresh, independent key pk' to encrypt this information. The evaluator will have to do some recrypting before he is able to use this information, but otherwise using independent keys does not complicate the construction much. More details on how the evaluation procedure deals with the different keys are provided in Protocol 6.3.2.

The size of the quantum sate $\gamma_{x,z}\left(g(sk)\right)$ depends on the garden-hose complexity of the classical decryption function HE.Dec. If it is in LOGSPACE (as most are), then Theorem 6.2.1 states that the size of the state is polynomial:

Lemma 6.2.3. *If* HE.Dec *is computable by a Turing machine that uses space $O(\log\lambda)$, where λ is the security parameter, then the number of qubits in $\gamma_{x,z}\left(g(sk)\right)$ is polynomial in λ.*

When Alice (the evaluator) is ready to use the gadget, she uses the value $\tilde{a}$ to compute her garden-hose strategy, a list of m disjoint measurement pairs of elements in $\{0,1,2,\ldots,2m\}$. The labels 1 through $2m$ refer to the qubits that make up the gadget (the "pipes"), and 0 is the label of the input qubit with the possible phase error. All but a single qubit will be measured: the remaining qubit will be the corrected qubit in the state $\mathsf{X}^{a'}\mathsf{Z}^{b'}\left|\psi\right\rangle$. We will call the (classical) function that computes the list of Alice's Bell measurements $\mathsf{TP.GenMeasurement}(\tilde{a})$.

Intuitively, after the evaluator has performed the measurements, the "path" the qubit has taken through the gadget state includes one of the pairs with an inverse phase gate whenever $\mathsf{HE.Dec}_{sk}(\widetilde{a}) = 1$, and avoids all such pairs when $\mathsf{HE.Dec}_{sk}(\widetilde{a}) = 0$.

6.2.4 Key update

After using the gadget, the evaluator has the following classical information: encrypted initial one-time pad keys $(\widetilde{a}, \widetilde{b})$, the encrypted gadget structure $\widetilde{g(sk)}$, the key generator's encrypted measurement outcomes $(\widetilde{x}, \widetilde{z})$, a list of Bell measurements from $\mathsf{TP.GenMeasurement}(\widetilde{a})$, and a list of outcomes of these m measurements, which we call $c, d \in \{0,1\}^m$. We sketch how the evaluator can homomorphically compute encryptions of the new keys a' and b'. Using those ciphertexts, the evaluator can continue the homomorphic quantum computation.

The first step is to encrypt the Bell measurements and their outcomes, so that all classical information is encrypted. For now, we will ignore the fact that different pieces of information may be encrypted under different public keys: later (Equation (6.10)), we will see that the evaluator is able to "recrypt" some pieces of information so that everything is encrypted under the same key. Once we are in that situation, the evaluator can homomorphically compute the update function, which we describe now.

The update function tracks the path of the qubit through the gadget, resolving the teleportations that involve the qubit one by one. Even though the measurements were not performed in this order, we can consider them to be for the sake of the teleportation corrections.

Let $\mathbf{a}, \mathbf{b}$ be variables that hold the current key to the one-time pad at every step of the algorithm: they are initialized as $\mathbf{a} \leftarrow a$ and $\mathbf{b} \leftarrow b$. Let $\mathbf{q} \in \{0,1\}$ be the variable that stores whether or not the qubit currently has an extra phase gate, initialized as $\mathbf{q} \leftarrow a$. Let $\mathbf{r} \in \{0,1,2,\dots,2m\}$ be the variable that contains the current location of the qubit, initialized as $\mathbf{r} \leftarrow 0$. That is, we view the current state as being $\mathsf{P}^{\mathbf{q}}\mathsf{X}^{\mathbf{a}}\mathsf{Z}^{\mathbf{b}}|\psi\rangle$ at location $\mathbf{r}$. These values are updated step by step.

In every odd step, we find the pair in $\mathsf{TP.GenMeasurement}(\widetilde{a})$ of the form $(\mathbf{r}, s)$ or $(s, \mathbf{r})$ for some other location s. If this is the ith pair, the measurement outcome is given by $c[i]$ and $d[i]$. This teleportation step changes the current state to

$$\mathsf{X}^{c[i]}\mathsf{Z}^{d[i]}\mathsf{P}^{\mathbf{q}}\mathsf{X}^{\mathbf{a}}\mathsf{Z}^{\mathbf{b}}|\psi\rangle = \mathsf{P}^{\mathbf{q}}\mathsf{X}^{\mathbf{a}\oplus c[i]}\mathsf{Z}^{\mathbf{b}\oplus d[i]\oplus \mathbf{q}\cdot c[i]}|\psi\rangle, \tag{6.6}$$

so we update the values as $\mathbf{a} \leftarrow \mathbf{a} \oplus c[i]$, $\mathbf{b} \leftarrow \mathbf{b} \oplus d[i] \oplus \mathbf{q} \cdot c[i]$, and $\mathbf{r} \leftarrow s$. Note that the key update contains a multiplication here, which has to be performed homomorphically.

In every even step, we do the same thing, except that we look for the pair $(\mathbf{r}, s)$ or $(s, \mathbf{r})$ in the list in $g(sk)$, and for the measurement outcome in the strings x, z. Additionally, we check whether an inverse phase gate was applied in the string p. The quantum state after the teleportation step equals

$$\mathsf{X}^{x[j]}\mathsf{Z}^{z[j]}(\mathsf{P}^{\dagger})^{p[j]}\mathsf{P}^{\mathbf{q}}\mathsf{X}^{\mathbf{a}}\mathsf{Z}^{\mathbf{b}}|\psi\rangle = \mathsf{X}^{x[j]}\mathsf{Z}^{z[j]}\mathsf{P}^{p[j]}\mathsf{P}^{\mathbf{q}}\mathsf{X}^{\mathbf{a}}\mathsf{Z}^{\mathbf{b}+p[j]}|\psi\rangle \tag{6.7}$$

$$= \mathsf{P}^{p[j]+\mathbf{q} \pmod 2}\mathsf{X}^{\mathbf{a}+x[j]}\mathsf{Z}^{\mathbf{b}+z[j]+p[j]\cdot(1+\mathbf{q})+x[j]\cdot(p[j]+\mathbf{q})}\left|\psi\right\rangle. \qquad (6.8)$$

For rewriting, we used the fact that $\mathsf{P}^2 = \mathsf{Z}$ and that $\mathsf{P}^\dagger = \mathsf{PZ}$, together with the commutation relations $\mathsf{ZP} = \mathsf{PZ}$ and $\mathsf{XP} = \mathsf{PXZ}$. We therefore update $\mathbf{q} \leftarrow p[j] + \mathbf{q} \pmod 2$, $\mathbf{a} \leftarrow \mathbf{a} + x[j]$, $\mathbf{b} \leftarrow \mathbf{b} + z[j] + p[j]\cdot(1+q[j]) + x[j]\cdot(p[j]+q[j])$, and $\mathbf{r} \leftarrow s$.

The path of the qubit is traced through a total of $2m$ steps. The final one-time-pad keys a' and b' equal the value of $\mathbf{a}$ and $\mathbf{b}$ after the last step.

6.3 TP: a fully homomorphic scheme

Our scheme TP (for teleportation) is an extension of the scheme CL (see Section 5.3): the quantum state is encrypted using a quantum one-time pad, and Clifford gates are evaluated simply by performing the gate on the encrypted state and then homomorphically updating the encrypted keys to the pad. Our new scheme TP, like AUX, includes additional resource states (the T-gate gadgets from Section 6.2) in the evaluation key. These gadgets can be used to immediately correct any P errors that might be present after the application of a T gate. The size of the evaluation key thus grows linearly with the upper bound t on the number of T gates in the circuit: for every T gate, the evaluation key contains one gadget, along with some classical information on how to use that gadget.

Key generation

The gadgets are generated during the key generation phase. Because the gadget structure depends on the secret key, we encrypt the classical information describing each gadget under a fresh public key. Using the classical HE.KeyGen as a subroutine to create multiple classical homomorphic key sets, we generate a classical secret and public key, and a classical-quantum evaluation key that contains t gadgets, allowing evaluation of a circuit containing up to t T gates. Every gadget depends on a different secret key, and its classical information is always encrypted using the next public key.

The key-generation procedure gets an extra parameter, t, which describes the number of T-gate gadgets it should produce. Since t gives an upper bound on the number of T gates that can be evaluated, the resulting scheme TP is leveled.

Protocol 6.3.1 (TP.KeyGen$(1^\lambda, 1^t)$)**.** Given inputs 1^λ (for a security parameter λ) and 1^t (for an upper bound on the number of T gates t), do the following:

1. For $i = 0$ to t, execute $(pk_i, sk_i, evk_i) \leftarrow \mathsf{HE.KeyGen}(1^\lambda)$ to obtain $t+1$ independent classical homomorphic key sets.
2. Set the public key to be the tuple $(pk_i)_{i=0}^{t}$.

3. Set the secret key to be sk_t.
4. For $i = 1$ to t: create the gadget $\Gamma_{pk_i}(sk_{i-1})$ as described in Section 6.2.3.
5. Set the evaluation key to be the set of all gadgets created in the previous step (including their encrypted classical information), plus the tuple $(evk_i)_{i=0}^t$. The resulting evaluation key is the quantum state

$$\bigotimes_{i=1}^{t} \Gamma_{pk_i}(sk_{i-1}) \otimes \bigotimes_{i=0}^{t} |evk_i\rangle\langle evk_i|. \tag{6.9}$$

To deal with multiple classical key sets (pk_i, sk_i, evk_i), we use the notation $\widetilde{x}^{[i]}$ to make it clear under which key the value x is encrypted. It is important to note that (e.g.) pk_i does *not* refer to the ith bit of the public key, but to the ith public key: in case we want to refer to the ith bit of some string s, we use the notation $s[i]$.

When working with multiple key sets, it will often be necessary to transform an already encrypted message $\widetilde{x}^{[i]}$ into an encryption $\widetilde{x}^{[j]}$ using a different key set $j \neq i$. We define the completely classical procedure $\mathsf{HE.Rec}_{i\to j}$ that can always be used for this *recryption* task as long as we have access to an encrypted version $\widetilde{sk_i}^{[j]}$ of the old secret key sk_i. Effectively, $\mathsf{HE.Rec}_{i\to j}$ homomorphically evaluates the decryption of $\widetilde{x}^{[i]}$:

$$\mathsf{HE.Rec}_{i\to j}(\widetilde{x}^{[i]}) := \mathsf{HE.Eval}_{evk_j}^{\mathsf{HE.Dec}}\left(\widetilde{sk_i}^{[j]}, \mathsf{HE.Enc}_{pk_j}(\widetilde{x}^{[i]})\right). \tag{6.10}$$

Encryption

The encryption procedure $\mathsf{TP.Enc}$ is identical to $\mathsf{CL.Enc}$ (see Protocol 5.3.1), using the first public key pk_0 for the encryption of the one-time-pad keys. We restate it here for completeness.

Every input qubit is encrypted separately with a quantum one-time pad, and the pad key is (classically) encrypted and appended to the quantum encryption:

$$\rho \mapsto \frac{1}{4} \sum_{a,b\in\{0,1\}} \mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a \otimes \left|\mathsf{HE.Enc}_{pk_0}(a,b)\right\rangle\!\left\langle\mathsf{HE.Enc}_{pk_0}(a,b)\right|. \tag{6.11}$$

Evaluation

Like CL, evaluation proceeds gate-by-gate. For measurements and Clifford gates, the evaluation procedure is identical to $\mathsf{CL.Eval}$ described in Protocol 5.3.1. At any point during the computation, the evaluator knows under which public key pk_i the classical information about the one-time pad is encrypted (initially, $i = 0$). She updates the keys homomorphically under pk_i. After a Clifford-gate evaluation, the classical information is encrypted under the same public key pk_i.

Before the evaluation of the ith T gate, the classical information is encrypted under pk_{i-1}. During the evaluation, the gadget $\Gamma_{pk_i}(sk_{i-1})$ is consumed. Afterwards, all classical information is encrypted under the *next* public key pk_i.

Protocol 6.3.2 ($\mathsf{TP.Eval}^{\mathsf{T}}$)**.** This protocol describes the execution of the ith T gate. Given a qubit $\mathsf{X}^a\mathsf{Z}^b\rho\mathsf{Z}^b\mathsf{X}^a$, encrypted keys $\widetilde{a}^{[i-1]}$, $\widetilde{b}^{[i-1]}$, and a gadget $\Gamma_{pk_i}(sk_{i-1})$, do the following:

1. Apply T to the qubit to obtain the state $\mathsf{P}^a\mathsf{X}^a\mathsf{Z}^b\mathsf{T}\rho\mathsf{T}^\dagger\mathsf{Z}^b\mathsf{X}^a\left(P^\dagger\right)^a$.
2. Run $M \leftarrow \mathsf{TP.GenMeasurement}(\widetilde{a}^{[i-1]})$ to obtain a list M of measurement instructions for the garden-hose gadget (see Section 6.2.3). Perform the Bell measurements, yielding outcomes $c, d \in \{0,1\}^m$ (where m is the number of measurements).
3. Compute the encryptions $\widetilde{c}^{[i]}, \widetilde{d}^{[i]} \leftarrow \mathsf{HE.Enc}_{pk_i}(c, d)$.
4. Recrypt the one-time-pad keys a and b into $\widetilde{a}^{[i]}$, $\widetilde{b}^{[i]}$, using the ciphertext $\widetilde{sk_{i-1}}^{[i]}$ contained in $\Gamma_{pk_i}(sk_{i-1})$.
5. Using classical information encrypted under pk_i, homomorphically compute the updated keys $\widetilde{a'}^{[i]}, \widetilde{b'}^{[i]}$, as described in Section 6.2.4.

At the end of the evaluation of some circuit c containing $i \leq t$ T gates, the evaluator holds a one-time-pad encryption of $\Phi_c\left(\rho\right)$, together with the keys to the pad, classically encrypted under the ith key. The last step is to recrypt (in $t - i$ steps) this classical information into the tth (final) key.

Decryption

The decryption procedure is identical to $\mathsf{CL.Dec}$. For each qubit, $\mathsf{HE.Dec}_{sk_t}$ is run twice in order to retrieve the keys to the quantum pad. The correct Pauli operator can then be applied to the quantum state (or only the X correction for measurement outcomes) in order to obtain the desired output state $\Phi_c\left(\rho\right)$.

6.3.1 Correctness and compactness

Correctness of the scheme TP follows from correctness of the evaluation procedures for the individual gates are. For Clifford gates and measurements, the argument is the same as for the scheme CL. For the T gate, correctness relies on the correctness of the garden-hose protocol underlying the T-gate gadget. If the garden-hose protocol is correct, the phase error is always correctly removed, as argued in Section 6.2.

Furthermore, since the decryption procedure is fairly straightforward, we can show that TP is compact:

Lemma 6.3.3. *If* HE *is compact, then* TP *is compact.*

Proof. By compactness of HE, there exists a polynomial $p(\lambda)$ such that for any function f, the complexity of applying HE.Dec to the output of $\mathsf{HE.Eval}^f$ is at most $p(\lambda)$. Since the keys to the quantum one-time pad of any wire are two single bits (a, b) encrypted with the classical HE scheme, decrypting the keys for one wire requires at most $2p(\lambda)$ steps. Obtaining the qubit then takes at most two steps more for (conditionally) applying X^a and Z^b. The classical and quantum decryption are repeated individually for every wire. The total number of steps is polynomial in λ and independent of the quantum circuit c, so we conclude that TP is compact. □

Although TP is compact in the sense of Definition 5.2.4, there is an important footnote to place here: in a delegated-computation setting, the amount of work that the client has to do is still dependent on the size of the circuit, because the client has to create (quantum) gadgets of significant size for every T gate. However, TP still has two main advantages over the trivial noncompact scheme (where the client performs all the computation during decryption).

First, the creation of the T-gate gadgets does not require a universal quantum computer. In particular, the client does not need the ability to perform T gates. In Section 6.6.2, we discuss the client's resources in more detail.

Second, moving the computational burden from decryption to key generation makes it independent of the input data. Keeping the decryption simple gives it more potential to be used in applications such as obfuscation [BR14]. In fact, in Chapter 7, the simplicity of decryption will be crucial to constructing a circuit that cannot be obfuscated.

6.3.2 Security

In this section, we show that TP is q-IND-CPA secure, i.e., no polynomial-time quantum adversary can tell the difference between an encryption of a real message and an encryption of $|0\rangle\langle 0|$, except with probability negligible in the security parameter (see Definition 5.2.9). Similarly to the proof sketch in Section 5.3 and the proof of Lemma 5.6.11, we use a reduction argument to relate the probability of distinguishing between two encryptions to the probability of distinguishing two classical encryptions under the scheme HE, which is known to be small.

The reduction argument, repeated t times, allows us to replace gadget states (containing sensitive information about the secret key) with fully mixed states, one by one. Doing so upper bounds the security error for TP by the error of that scheme with a fully mixed evaluation key: essentially, we will have arrived at the Clifford scheme CL, which is already known to be q-IND-CPA secure whenever HE is.

We start by defining a sequence of variations on the TP scheme. For $\ell \in \{0,\ldots,t\}$, let $\mathsf{TP}^{(\ell)}$ be identical to TP, except for the key-generation procedure: $\mathsf{TP}^{(\ell)}.\mathsf{KeyGen}$ replaces, for every $i > \ell$, all classical information accompanying the ith gadget with the all-zero string before encrypting it.

Write $g_i := g(sk_{i-1})$ for the classical information accompanying the ith gadget. Then $\mathsf{TP}^{(\ell)}.\mathsf{KeyGen}(1^\lambda, 1^t)$ outputs

$$\bigotimes_{i=0}^{t} |evk_i\rangle\langle evk_i| \otimes \bigotimes_{i=1}^{\ell} \Gamma_{pk_i}(sk_{i-1}) \otimes$$
$$\bigotimes_{i=\ell+1}^{t} \Big(\big|\mathsf{HE.Enc}_{pk_i}(0^{|g_i|})\big\rangle\big\langle \mathsf{HE.Enc}_{pk_i}(0^{|g_i|})\big| \otimes$$
$$\frac{1}{2^{2m}} \sum_{x_i,z_i \in \{0,1\}^m} \big|\mathsf{HE.Enc}_{pk_i}(0^m,0^m)\big\rangle\big\langle \mathsf{HE.Enc}_{pk_i}(0^m,0^m)\big| \otimes \gamma_{x_i,z_i}(g_i)\Big). \quad (6.12)$$

As noted in Section 6.2.3, the length of the classical information g_i does not depend on sk_{i-1} itself, so a potential adversary cannot gain any information about sk_{i-1} just from this encrypted string.

Intuitively, one can view $\mathsf{TP}^{(\ell)}$ as the scheme that provides only ℓ usable gadgets in the evaluation key. Note that $\mathsf{TP}^{(t)} = \mathsf{TP}$. In $\mathsf{TP}^{(0)}$, only the classical evaluation keys remain, together with encryptions of zeros and the completely mixed state instead of $\gamma_{x_i,z_i}(g_i)$ (cf. Equation (6.4)).

As a step towards the security of TP, we show that in the quantum CPA indistinguishability experiment, any efficient adversary interacting with $\mathsf{TP}^{(\ell)}$ only has negligible advantage with respect to the same experiment interacting with $\mathsf{TP}^{(\ell-1)}$. That is, the encrypted classical information g_ℓ, x_ℓ, z_ℓ does not give a significant advantage.

Lemma 6.3.4. *If* HE *is q-IND-CPA secure, then for any quantum polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, *there exists a negligible function* negl *such that for all* $1 \leq \ell \leq t$,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda) = 1] - \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda) = 1] \leq \mathrm{negl}(\lambda).$$

Proof. The difference between schemes $\mathsf{TP}^{(\ell)}$ and $\mathsf{TP}^{(\ell-1)}$ lies in whether the gadget state $\gamma_{x_\ell,z_\ell}(g_\ell)$ is supplemented with its classical information $\widetilde{g_\ell}, \widetilde{x_\ell}, \widetilde{z_\ell}$, or just with an encryption of $0^{|g_\ell|+2m}$.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary for the game $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda)$ for any ℓ (note that the input/output structure is the same for all games). We will define an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$ that, for a random choice of ℓ, either simulates the game $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda)$ or $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda)$. Which game is simulated will depend on some $s \in_R \{0,1\}$ that is unknown to $\mathcal{A}'$ herself. Using the assumption that HE is q-IND-CPA secure, we are able to argue that $\mathcal{A}'$ cannot recognize which of the two schemes was simulated, and this fact allows us to bound the difference in success probabilities between the security games of $\mathsf{TP}^{(\ell)}$ and $\mathsf{TP}^{(\ell-1)}$. Crucially, because $\mathcal{A}'$ does not depend on ℓ (it only selects a random ℓ internally), the negligible function negl will also be independent of ℓ. Such a randomizing argument is a standard technique to get rid of the dependence on ℓ [KL14].

The adversary $\mathcal{A}'$ acts as follows (see also Figure 6.5):

$\mathcal{A}'_1$ selects a random value $\ell \in [t]$. She then starts simulating the indistinguishability game using $\mathcal{A}$ as an adversary. $\mathcal{A}'_1$ takes care of most of the key-generation procedure: she generates the classical key sets 0 through $\ell - 1$ herself, generates the random strings $x_1, z_1, \dots, x_\ell, z_\ell$, and then constructs the gadgets $\gamma_{x_1,z_1}(g_1), \dots, \gamma_{x_\ell,z_\ell}(g_\ell)$ and their classical information $g_1, \dots, g_\ell$. She encrypts the classical information using the appropriate public keys. Only g_ℓ, x_ℓ and z_ℓ are left unencrypted: instead of encrypting these strings herself using a generated key pk_ℓ, $\mathcal{A}'_1$ sends the strings for encryption to the challenger. Whether the challenger really encrypts g_ℓ, x_ℓ and z_ℓ, or replaces the strings with a string of zeros, determines which of the two schemes is simulated ($\mathsf{TP}^{(\ell)}$ or $\mathsf{TP}^{(\ell-1)}$). $\mathcal{A}'$ is unaware of the random choice s of the challenger.

The adversary $\mathcal{A}'_1$ also generates the extra padding inputs that correspond to the already-removed gadgets $\ell + 1$ up to t. Since these gadgets consist of all-zero strings encrypted with independently chosen public keys that are not used anywhere else, together with a completely mixed quantum state, the adversary can generate them without needing any extra information.

$\mathcal{A}'_2$ feeds the evaluation key and public key, just generated by $\mathcal{A}'_1$, to $\mathcal{A}_1$ in order to obtain a chosen message in the register X (plus the auxiliary register R). She then picks a random $r \in_R \{0,1\}$ and erases the state in X if and only if $r = 0$. She encrypts the result according to the $\mathsf{TP.Enc}$ procedure (using the public key $(pk_i)_{i=0}^t$ received from $\mathcal{A}'_1$), and gives the encrypted state, plus R, to $\mathcal{A}_2$, who outputs r' in an attempt to guess r. $\mathcal{A}'_2$ outputs $s' = 1$ if and only if the guess by $\mathcal{A}$ was correct, i.e., $r' = r$.

Because HE is q-IND-CPA secure, the probability that $\mathcal{A}'$ wins $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$, i.e., that $s' = s$, is at most $1/2 + \mathrm{negl}(\lambda)$. For each choice of ℓ, there are two scenarios in which $\mathcal{A}'$ wins the game:

- $s = 1$ *and* $\mathcal{A}$ guesses r correctly: If $s = 1$, the game that is being simulated is $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda)$. If $\mathcal{A}$ wins the simulated game ($r = r'$), then $\mathcal{A}'$ will correctly output $s' = 1$. (If $\mathcal{A}$ loses, then $\mathcal{A}'$ outputs 0, and loses as well).
- $s = 0$ *and* $\mathcal{A}$ does not guess r correctly: If $s = 0$, the game that is being simulated is $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda)$. If $\mathcal{A}$ loses the game ($r \neq r'$), then $\mathcal{A}'$ will correctly output $s' = 0$. (If $\mathcal{A}$ wins, then $\mathcal{A}'$ outputs 1 and loses).

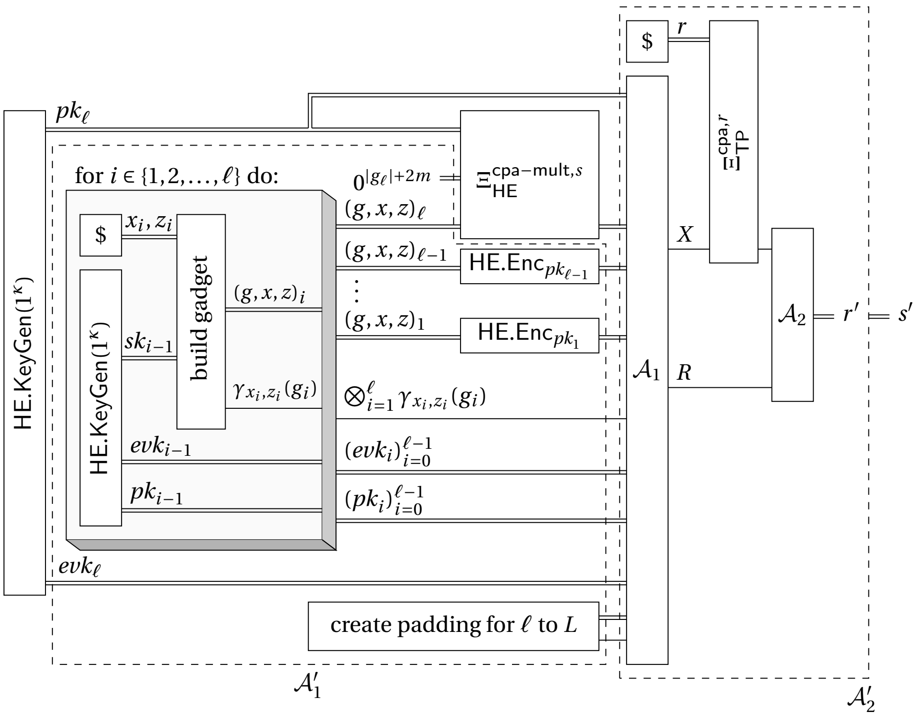


Figure 6.5: A strategy for the game $\mathsf{PubK}^{\mathsf{cpa-mult}}_{\mathcal{A}',\mathsf{HE}}(\lambda)$, using an adversary $\mathcal{A}$ for $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda)$ as a subroutine. All the wires going into $\mathcal{A}_1$ together form the evaluation key and public key for $\mathsf{TP}^{(\ell)}$ or $\mathsf{TP}^{(\ell-1)}$, depending on s. Note that $\Xi^{\mathsf{cpa},r}_{\mathsf{TP}} = \Xi^{\mathsf{cpa},r}_{\mathsf{TP}^{(\ell)}} = \Xi^{\mathsf{cpa},r}_{\mathsf{TP}^{(\ell-1)}}$, so $\mathcal{A}'_2$ can run either one of these independently of s (i.e., without having to query the challenger). The “create padding” subroutine generates dummy gadgets for $\ell+1$ up to t, as described in the definition of $\mathcal{A}'_1$.

From the above, we conclude that the winning probability of $\mathcal{A}'$ is

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\ell \in_R [t]} \left(\Pr[s=1] \cdot \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda) = 1] + \Pr[s=0] \cdot \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda) = 0] \right) && (6.13)\\
&\quad \leqslant \frac{1}{2} + \mathrm{negl}(\lambda), && (6.14)
\end{aligned}$$

from which it follows that

$$\sum_{\ell=1}^{t} \frac{1}{t} \left(\frac{1}{2} \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda) = 1] + \frac{1}{2} \left(1 - \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda) = 1] \right) \right) \leqslant \frac{1}{2} + \mathrm{negl}(\lambda), \tag{6.15}$$

and therefore, for all $\ell \in [t]$,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell)}}(\lambda) = 1] - \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(\ell-1)}}(\lambda) = 1] \leqslant 2t \cdot \mathrm{negl}(\lambda). \tag{6.16}$$

As long as t, the number of t gates in the circuit, is polynomial in λ, the bound is still negligible (and independent of ℓ). $\square$

By applying Lemma 6.3.4 iteratively, t times in total, we are able to conclude that the advantage of an adversary interacting with $\mathsf{TP}^{(t)}$ over one interacting with $\mathsf{TP}^{(0)}$ is also negligible:

Corollary 6.3.5. *If t is polynomial in λ, then for any quantum polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(t)}}(\lambda) = 1] - \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(0)}}(\lambda) = 1] \leqslant \mathrm{negl}(\lambda).$$

For Corollary 6.3.5, it is important that the negligible difference between the winning probabilities for $\mathsf{TP}^{(\ell)}$ and $\mathsf{TP}^{(\ell-1)}$ is independent of ℓ (see Lemma 6.3.4). Then we can conclude that this negligible function, multiplied by a polynomial, is still negligible.

Using Corollary 6.3.5, we can finally state and prove the main result of this section:

Theorem 6.3.6. *If HE is q-IND-CPA secure, then TP is q-IND-CPA secure for circuits containing up to polynomially (in λ) many T gates.*

Proof. The scheme $\mathsf{TP}^{(0)}$ is very similar to CL in terms of its key generation and encryption steps. The evaluation key consists of several classical evaluation keys, plus some completely mixed states and encryptions of 0 which we can safely ignore because they are not related to the encrypted message. In both schemes, the encryption of a qubit is a quantum one-time pad together with the encrypted keys. The only difference is that in $\mathsf{TP}^{(0)}$, the public key and evaluation key form a tuple containing a list of public/evaluation keys that are independent of the encryption (in addition to

pk_0 and evk_0 which *are* used for the encryption of the quantum one-time pad). These keys do not provide any advantage (in fact, the adversary could have generated them herself by repeatedly running $\mathsf{HE.KeyGen}(1^\lambda, 1^t)$). Therefore, we can safely ignore these keys as well.

Because of the similarity between CL and $\mathsf{TP}^{(0)}$, the exact same proof as in Section 5.3 shows that $\mathsf{TP}^{(0)}$ is q-IND-CPA secure. That is, for any $\mathcal{A}$,

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(0)}}(\lambda) = 1] \leq \frac{1}{2} + \operatorname{negl}(\lambda). \tag{6.17}$$

Combining this result with Corollary 6.3.5, it follows that

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}}(\lambda) = 1] &\leq \Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathsf{TP}^{(0)}}(\lambda) = 1] + \operatorname{negl}(\lambda) &(6.18)\\ &\leq \frac{1}{2} + \operatorname{negl}(\lambda) + \operatorname{negl}(\lambda). &(6.19)\end{aligned}$$

Since the sum of two negligible functions is itself negligible, the statement follows. □

6.3.3 Circuit privacy

As argued in Section 5.3.1, the Clifford scheme CL can straightforwardly be adapted to provide circuit privacy in the semi-honest setting. The same adaptation (randomizing the output data with an additional quantum one-time pad before returning the output to the client) will give circuit privacy to TP as well.

To show circuit privacy, we need to argue that the recryptions into different classical key sets do not degrade the privacy of the computation:

Lemma 6.3.7. *Suppose* HE *has statistical circuit privacy in the semi-honest setting, and let the simulator* $\mathcal{S}_{\mathsf{HE}}$ *be as in Definition 5.2.6. Then for any security parameter* λ*,* t *polynomial in* λ*, list of classical functions* $f_0, f_1, \ldots, f_t$*, list of key sets* $(pk_i, evk_i, sk_i)_{i=1}^t$ *generated by* $\mathsf{HE.KeyGen}(1^\lambda)$*, and input* x*, the statistical distance between*

$$\mathsf{HE.Eval}^{f_t}_{evk_t}\left(\mathsf{HE.Rec}_{(t-1)\to t}(\cdots \mathsf{HE.Eval}^{f_1}_{evk_1}(\mathsf{HE.Rec}_{0\to 1}(\mathsf{HE.Eval}^{f_0}_{evk_0}(\mathsf{HE.Enc}_{pk_0}(x)))))\right)$$

and

$$\mathcal{S}_{\mathsf{HE}}\left(1^\lambda, pk_t, evk_t, f_t(\cdots f_1(f_0(x)))\right)$$

is negligible in λ*.*

Proof. Since $\mathsf{HE.Rec}_{(t-1)\to t} = \mathsf{HE.Eval}^{\mathsf{HE.Dec}_{sk_{t-1}}}_{evk_t} \circ \mathsf{HE.Enc}_{pk_t}$ by definition, we have that

$$\mathsf{HE.Eval}^{f_t}_{evk_t}\left(\mathsf{HE.Rec}_{(t-1)\to t}(\cdots (\mathsf{HE.Enc}_{pk_0}(x)))\right) \tag{6.20}$$

$$= \quad \mathsf{HE.Eval}_{evk_t}^{f_t \circ \mathsf{HE.Dec}_{sk_{t-1}}}\left(\mathsf{HE.Enc}_{pk_t}(\cdots(\mathsf{HE.Enc}_{pk_0}(x)))\right) \tag{6.21}$$

$$\approx_{\text{negl}(\lambda)} \quad \mathcal{S}_{\mathsf{HE}}\left(1^\lambda, pk_t, evk_t, f_t(\mathsf{HE.Dec}_{sk_{t-1}}(\cdots(\mathsf{HE.Enc}_{pk_0}(x))))\right), \tag{6.22}$$

which, by correctness of HE, is statistically indistinguishable from

$$\mathcal{S}_{\mathsf{HE}}\left(1^\lambda, pk_t, evk_t, f_t(f_{t-1}(f_{t-2}(\cdots f_1(x))))\right) \tag{6.23}$$

as long as t is polynomial in λ. By the triangle inequality, the statement of the lemma follows. □

Using the above lemma, we can show that the adapted scheme TP′ (which adds an additional quantum one-time pad at the end of the evaluation) has circuit privacy whenever the underlying classical scheme does.

Lemma 6.3.8. *If* HE *has circuit privacy in the semi-honest setting, then* TP′ *does too.*

Proof. The proof is identical to the proof of Theorem 5.3.2. For the approximation step from Equation (5.14) to Equation (5.15), we use Lemma 6.3.7. □

6.4 TTP: a fully homomorphic scheme with verification

In this section, we introduce our candidate scheme for verifiable quantum fully homomorphic encryption (vQFHE), which integrates the T-gate gadget into the verifiable Clifford scheme TCL (see Section 5.6). We will define the scheme, and prove its correctness, compactness, privacy, and secure verifiability. In many ways, TTP is similar to TCL: where appropriate, we will refer back to Section 5.6 for details on the definitions and proofs.

The setup (underlying CSS code, classical MAC, and classical FHE scheme) is the same as for TCL, with one difference: we now require that the classical homomorphic-encryption scheme has decryption in LOGSPACE, to ensure that the T-gate gadgets remain polynomial.

Key generation

In addition to encoded magic states for P and H, the key generation will produce encoded magic states for T, and *encoded* error-correction gadgets (see Section 6.2).

For the gadget state $\gamma_{x,z}$, we can assume that the qubits labeled "in" and "out" are independent of the inputs sk and $\widetilde{a}$. This property can be enforced at the cost of an extra constant number of qubits in the gadget, as follows. In the garden-hose game, we add two extra pipes at the start: the first pipe is *always* connected to the tap by Alice, and connected to the second pipe by Bob. Alice can then connect the second pipe to whichever pipe she would normally have connected the tap to. For the "out" qubit, we can use the same construction, adding two extra pipes so that the water always flows out of the same location. In the garden-hose gadget, this alteration amounts to two extra EPR pairs: one for "in", one for "out".

The above assumption allows us to consistently break a gadget state γ into three parts: γ^{in} (the single input qubit), γ^{mid}, and γ^{out} (the single output qubit). We will write $\mathsf{TTP.GadgetGen}$ for the algorithm that, given a secret key sk, generates the quantum gadget $(\gamma^{\mathsf{in}}, \gamma^{\mathsf{mid}}, \gamma^{\mathsf{out}})$ plus the accompanying classical information $g(sk)$, according to the specifications in Section 6.2.3. To create the full gadget state $\Gamma_{pk'}(sk)$ from this, one would have to encrypt $(\gamma^{\mathsf{in}}, \gamma^{\mathsf{mid}}, \gamma^{\mathsf{out}})$ with a one-time pad, and encrypt the pad keys and $g(sk)$ with a classical FHE scheme. In TTP, this encryption will work differently, because we want to add authentication (in the form of the trap code and a classical MAC). The qubits in γ^{mid} will be encrypted using different (independent) permutations from all other states in $\mathsf{TTP.KeyGen}$.

Protocol 6.4.1 ($\mathsf{TTP.KeyGen}$)**.** Given a security parameter λ and gate parameters p, h, t, do the following:

1. Sample classical keys as

$$k \leftarrow \mathsf{MAC.KeyGen}, \tag{6.24}$$

$$\pi \leftarrow S_{3n}, \tag{6.25}$$

$$(sk_i, pk_i, evk_i) \leftarrow \mathsf{HE.KeyGen} \text{ (for } i = 0, \ldots, t). \tag{6.26}$$

2. Set $sk := (\pi, k, sk_0, \ldots, sk_t, pk_0)$ to be the secret key.
3. For $i \in [p]$, let $\mu_i^{\mathsf{P}} \leftarrow \mathsf{TTP.Enc}_{sk}(\mathsf{P}\,|+\rangle)$ be the encrypted magic state for P. For $\mathsf{TTP.Enc}$, see below.
4. For $i \in [h]$, let $\mu_i^{\mathsf{H}} \leftarrow \mathsf{TTP.Enc}_{sk}\left(\frac{1}{\sqrt{2}}(\mathsf{H} \otimes \mathsf{I})\,(|00\rangle + |11\rangle)\right)$ be the encrypted magic state for H.
5. For $i \in [t]$, let $\mu_i^{\mathsf{T}} \leftarrow \mathsf{TTP.Enc}_{sk}(\mathsf{T}\,|+\rangle)$ be the encrypted magic state for T.
6. For $i \in [t]$, create the error-correction gadget Γ_i as follows:
   - (a) Sample a fresh permutation $\pi_i \leftarrow_R S_{3n}$.
   - (b) Compute $(g_i, \gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}}) \leftarrow \mathsf{TTP.GadgetGen}(sk_{i-1})$, as described above.
   - (c) Compute $\Gamma_i \leftarrow \mathsf{TTP.Enc}_{sk}(\gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{out}}) \otimes \mathsf{TTP.Enc}_{(\pi_i, k, sk_0, \ldots, sk_t, pk_i)}(\gamma_i^{\mathsf{mid}}) \otimes \mathsf{MAC.Sign}_k(\mathsf{HE.Enc}_{pk_i}(g_i, \pi_i))$.

7. Let $keys \leftarrow \mathsf{MAC.Sign}_k(pk_0, \ldots, pk_t, evk_0, \ldots, evk_t, \mathsf{HE.Enc}_{pk_0}(\pi))$ be the authenticated list of keys. The secret information π is encrypted before it is signed.

8. Output the secret key $sk = (\pi, k, sk_0, \ldots, sk_t, pk_0)$, along with the quantum evaluation key $(keys, \mu_1^{\mathsf{P}}, \ldots, \mu_p^{\mathsf{P}}, \mu_1^{\mathsf{H}}, \ldots, \mu_h^{\mathsf{H}}, \mu_1^{\mathsf{T}}, \ldots, \mu_t^{\mathsf{T}}, \Gamma_1, \ldots, \Gamma_t)$.

Encryption

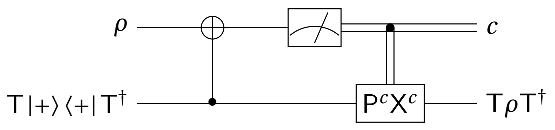
Encryption works in the same way as TCL.Enc (see Protocol 5.6.2): one-time pad keys are sampled, and the quantum state is encoded under the trap code using those keys. A signed encryption of the pad keys is appended. Note that by default, the classical public key pk_0 is used for encrypting the pad keys, given how the encryption key sk is defined in Protocol 6.4.1. During certain parts of the gadget generation, however, we give a different encryption key to TTP.Enc, containing the ith public key pk_i (instead of pk_0), and the independent permutation π_i (instead of the global permutation π). See Step 6c of Protocol 6.4.1.

Evaluation

The evaluation of Clifford gates and computational-basis measurements is the same as in the verifiable Clifford scheme TCL (see Protocols 5.6.3 to 5.6.6). The only difference is that, as in TP, the public key may switch. After the ith T gate, all classical information is recrypted into pk_i (initially, it is encrypted under pk_0). The evaluator always knows under which public key the computation is currently running.

The evaluation of a T gate requires an encrypted magic state and encrypted error-correction gadget as a resource, which are both part of the evaluation key.

Recall the circuit for magic-state computation of a T gate from Section 2.4.4:


ρ
$\mathsf{T}\lvert+\rangle\langle+\rvert\mathsf{T}^\dagger$
c
$\mathsf{P}^c\mathsf{X}^c$
$\mathsf{T}\rho\mathsf{T}^\dagger$

(6.27)

This circuit is much more complicated than the magic-state computation for a P or H gate, since it requires the application of a classically-controlled phase correction P, which is not a Pauli. We will accomplish this using the error-correction gadget Γ_i.

First, we remark on some subtleties regarding the encrypted classical information surrounding the gadget. Since the structure of Γ_i depends on the classical secret key sk_{i-1}, the classical information about Γ_i is encrypted under the (independent) public key pk_i (see Protocol 6.4.1). This observation will play a crucial role in our proof that TTP satisfies IND-VER, in Section 6.4.2.

The usage of different key sets also means that, at some point during the evaluation of a T gate, all classically encrypted information needs to be recrypted from the $(i-1)$st into the ith key set. This can be done because $\widetilde{sk}_{i-1}$ is included in the classical information g_i in Γ_i. See also Equation (6.10).

Protocol 6.4.2 ($\mathsf{TTP.Eval}^{\mathsf{T}}$)**.** Given a ciphertext qubit $\tilde{\sigma}$ (with encrypted one-time pad keys $\tilde{x}, \tilde{z}$), an encrypted T-magic state μ_i^{T}, a gadget Γ_i, and an evaluation key containing $\tilde{\pi}$, evk_{i-1}, evk_i, pk_{i-1}, pk_i, do the following:

1. Run $\mathsf{TTP.Eval}^{\mathsf{CNOT}}$ (see Protocol 5.6.6) to perform a CNOT between μ_i^{T} (control) and $\tilde{\sigma}$ (target). The classically encrypted information is under pk_{i-1}. Call the resulting qubits $\widetilde{\sigma_1}$ (control, with keys $\widetilde{x_1}^{[i-1]}$ and $\widetilde{z_1}^{[i-1]}$) and $\widetilde{\sigma_2}$ (target, with keys $\widetilde{x_2}^{[i-1]}$ and $\widetilde{z_2}^{[i-1]}$), and the classical computation log log_1.

2. Run $\mathsf{TTP.EvalMeasurement}$ (see Protocol 5.6.3) to perform a measurement on $\widetilde{\sigma_2}$. The classically encrypted information is still under pk_{i-1}. Call the encrypted measurement result $\widetilde{b}^{[i-1]}$, and the classical computation log log_2.

3. Recrypt all classically encrypted information (except $\widetilde{b}^{[i-1]}$) from key set $i-1$ into key set i. Call the classical computation log log_3.

4. Run $M \leftarrow \mathsf{TTP.GenMeasurement}(\widetilde{b}^{[i-1]})$ to obtain a list M of measurement instructions for the garden-hose gadget (see Section 6.2.3). Evaluate the Bell measurements (including the measurement between $\widetilde{\sigma_1}$ and the "tap" qubit γ_i^{in}) according to M. Note that these are not physical Bell measurements, but executions of $\mathsf{TTP.Eval}$ for a CNOT, computational-basis measurement, and Hadamard-basis measurement. Call the (encrypted) measurement results $\widetilde{c}^{[i]}, \widetilde{d}^{[i]} \in \{0,1\}^{3n\cdot m}$, where m is the number of measurements in the list M. Call the classical computation log log_4.

5. Homomorphically compute the updated keys $\widetilde{x'}^{[i]}, \widetilde{z'}^{[i]}$ for the qubit γ_i^{out} that now contains the data qubit, using the measurement outcomes $\widetilde{c}, \widetilde{d}$, and the structural information in $\widetilde{g(i)}$. The key updates happen as described in Section 6.2.4, but transversally for each physical qubit. Call the classical computation log log_5.

6. Output the remaining unmeasured qubit $(\widetilde{\gamma_i^{\mathsf{out}}})$, the updated one-time pad keys, and the computation log $(log_1, \ldots, log_5)$.

Decryption

Finally, verified decryption is the same as TCL.VerDec: some classical checks are performed (MAC-verification, internal consistency of the computation log, and consistency between the log and the claimed circuit), after which all unmeasured traps are checked and the remaining qubits are decoded. TTP.VerDec follows Protocol 5.6.7, except that HE.Dec uses the tth secret key sk_t to decode the one-time pad keys.

6.4.1 Correctness, compactness, and privacy

Correctness of TTP relies mostly on the correctness of the Clifford scheme TCL: after all, TTP is an extension of TCL (see Figure A on page 289). For the T gate, Clifford operations are (homomorphically) executed on an encoded magic state and correction gadget: correctness for this part of the protocol follows from the correct evaluation of those Clifford operations (i.e., correctness of TCL), the correctness of the garden-hose gadget (argued in Section 6.2), and the correctness of magic-state computation. The only detail that should be addressed is the fact that part of the gadget is encoded under a different permutation. To see that this different encoding does not pose a problem, note that no two-qubit gate is ever evaluated between qubits that are encoded under different permutations: the data qubit (encoded under the global permutation π) is teleported in via γ_i^{in} (also encoded under π), while the Bell measurements within γ_i^{mid} are all evaluated between qubits under the local permutation π_i. The trap code only requires the permutations on two logical qubits to match if a two-qubit gate is evaluated on those two qubits.

Compactness (as in Definition 5.4.3) is satisfied by the same argument as for TCL, since the definition of VerDec remains unchanged.

TTP provides privacy in the sense of q-IND-CPA security: like TCL, it satisfies the private-key version of Definition 5.2.9. This property can again be shown by reduction to the q-IND-CPA property of HE. For TTP, the reduction is slightly nontrivial since the structure of the error-correction gadgets depends on the classical secret key. Similarly to the proof of Lemma 6.3.4, the reduction can be done in steps, where first the security of the encryptions under pk_t is applied (no gadget depends on sk_t). After that, the quantum part of the gadget Γ_t (which depends on sk_{t-1}) looks completely mixed from the point of view of the adversary. Next, we can apply security of the encryptions under pk_{t-1}, and repeat the process. After all encryptions of the quantum-one-time-pad keys are removed, the encryption of a state appears fully mixed. We omit the details here.

6.4.2 Secure verifiability

To show that TTP is IND-VER (and therefore also SEM-VER), we will again rely on Game 5.6.8, the hybrid indistinguishability game (Figure 5.3). This time, we *will* utilize the quantum wire that runs from KeyGen to VerDec: it will serve to send (variations of) the error-correction gadgets directly to VerDec, bypassing the adversary.

Similarly to the proof of Lemma 6.3.4, we will remove the error-correction gadgets one by one, in t rounds total. This time, however, we need to be more careful, because we do not want to affect the acceptance probability of VerDec too much. Thus, we need to make sure to replace the sk_i-dependent gadget with a "gadget" that is independent of the secret key, but still removes the unwanted phase P.

Apart from removing the gadgets, the proof structure will be very similar to that of the verifiability of TCL: we will first remove the classical MAC, then the computation log, then the gadgets, and finally all classical FHE encryptions.

Hybrid 1: Removing the classical MAC

Define TTP' analogously to TCL': all information that is signed under the classical MAC (in KeyGen and Enc, including the gadget structure) is sent directly to VerDec via the classical side channels. VerDec uses this side-channel information instead of the signed information (but still checks the validity of the classical signatures it receives from the adversary).

Lemma 6.4.3. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}_{\mathsf{TTP}}^{\mathsf{TTP}'}(\mathcal{A},\lambda) \leq \mathrm{negl}(\lambda)$.

Proof. Identical to the proof of Lemma 5.6.9: by reduction to the fact that MAC is existentially unforgeable under chosen-message attacks (EUF-CMA). □

Hybrid 2: Removing the computation log

Define TTP'' analogously to TCL'': The one-time-pad decryption keys are computed by VerDec on a plaintext level, based on the initial encryption keys of the input state and the measurement results throughout the computation. VerDec still checks the consistency of the classical log, and whether the operations in the classical log match the claimed circuit.

Lemma 6.4.4. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}_{\mathsf{TTP}'}^{\mathsf{TTP}''}(\mathcal{A},\lambda) \leq \mathrm{negl}(\lambda)$.

Proof. Analogous to the proof of Lemma 5.6.10: by reduction to the correctness of HE. Because TTP'' uses $t+1$ different classical encryption keys, we need to generalize Equation (5.37) as follows:

$$\Pr\left[\mathsf{HE.Dec}_{sk_t}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_t}(\mathsf{HE.Enc}_{pk_0}(s))) \neq f(s)\right] \leq \mathrm{negl}(\lambda). \tag{6.28}$$

In the above expression, we slightly abuse notation and write $\mathsf{HE.Eval}_{evk_0,\ldots,evk_t}$ to include the t recryption steps that are performed during TTP.Eval. As long as the number of T gates, and thus the number of recryptions, is polynomial in λ, the expression holds. □

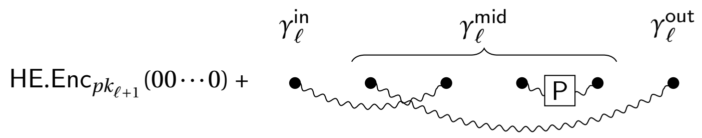

$\mathsf{HE.Enc}_{pk_{\ell+1}}(00\cdots0)$ +
$\gamma_\ell^{\mathsf{in}}$
$\gamma_\ell^{\mathsf{mid}}$
$\gamma_\ell^{\mathsf{out}}$
P

Figure 6.6: In $\mathsf{TTP}_1^{(\ell)}$, all classically encrypted information for the ℓth gadget is replaced by zeros. The quantum state remains the same as in TTP, depicted here as a simplified structure. The actual state $\gamma_\ell^{\mathsf{mid}}$ will be much larger than four qubits.

More hybrids: Removing gadgets

We continue by defining a sequence of hybrid schemes based on TTP''. In $4t$ steps, we will move all error-correction functionality from the gadgets to VerDec. Doing so will result in the adversary having no information about the classical secret keys (which are involved in constructing these gadgets). This will allow us to eventually reduce the security of TTP to that of TC.

We remove the gadgets back-to-front, starting with the final gadget. Every gadget is removed in four steps. For all $1 \leqslant \ell \leqslant t$, define the hybrids $\mathsf{TTP}_1^{(\ell)}$, $\mathsf{TTP}_2^{(\ell)}$, $\mathsf{TTP}_3^{(\ell)}$, and $\mathsf{TTP}_4^{(\ell)}$ (and $\mathsf{TTP}_4^{(t+1)} := \mathsf{TTP}''$) as follows:

1. $\mathsf{TTP}_1^{(\ell)}$ is the same as $\mathsf{TTP}_4^{(\ell+1)}$, except for the generation of the state Γ_ℓ (Step 6 of Protocol 6.4.1). In $\mathsf{TTP}_1^{(\ell)}$, all classical information encrypted under pk_ℓ is replaced with encryptions of zeros. In particular, for $i = \ell$, Step 6c is adapted to

$$\begin{aligned}\Gamma_i \leftarrow & \mathsf{TTP}''.\mathsf{Enc}_{sk}(\gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{out}}) \otimes \mathsf{TTP}''.\mathsf{Enc}'_{(\pi_i,k,sk_0,\ldots,sk_t,pk_i)}(\gamma_i^{\mathsf{mid}}) \otimes \\ & \mathsf{MAC.Sign}(\mathsf{HE.Enc}_{pk_i}(00\cdots0)),\end{aligned} \tag{6.29}$$

where $\mathsf{TTP}''.\mathsf{Enc}'$ also appends a signed encryption of zeros, that is, the encryption $\mathsf{TTP}''.\mathsf{Enc}'_{(\pi,k,sk_0,\ldots,sk_t,pk)}(\sigma)$ equals

$$\sum_{x,z\in\{0,1\}^{3m}} \big(\mathsf{TC.Enc}_{\pi,x,z}(\sigma) \otimes \mathsf{MAC.Sign}_k(\mathsf{HE.Enc}_{pk}(00\cdots0))\big). \tag{6.30}$$

See Figure 6.6 for a pictorial representation of the adapted gadget. There are two important remarks. First, the encryption of input states (and the gadgets $i < \ell$) is still performed using $\mathsf{TTP}''.\mathsf{Enc}$, so the classical information is still encrypted. Second, whenever classical information is removed in KeyGen or Enc', it *is* still sent to VerDec through the classical side channel. Hence, the structural and encryption information about Γ_ℓ is kept from the adversary, and instead is directly sent (only) to the verification procedure. Whenever VerDec needs this information, it is taken directly from this trusted source, and the all-zero string sent by the adversary will be ignored.

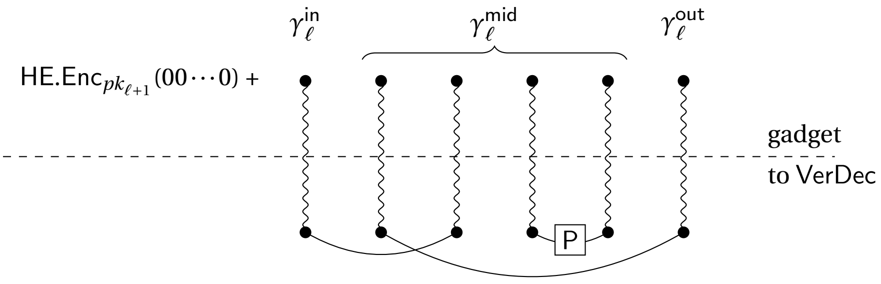

$\gamma_\ell^{\mathsf{in}}$
$\gamma_\ell^{\mathsf{mid}}$
$\gamma_\ell^{\mathsf{out}}$
$\mathsf{HE.Enc}_{pk_{\ell+1}}(00\cdots0)$ +
gadget
to VerDec
P

Figure 6.7: In $\mathsf{TTP}_2^{(\ell)}$, the quantum state that constitutes the ℓth gadget is replaced with halves of EPR pairs. The other halves are sent to VerDec, where Bell measurements and the phase gate P are applied after evaluation.

2. $\mathsf{TTP}_2^{(\ell)}$ is the same as $\mathsf{TTP}_1^{(\ell)}$, except that for the ℓth gadget, the procedure TTP.PostGadgetGen is called instead of TTP.GadgetGen (Step 6b of Protocol 6.4.1) (see also Figure 6.7):

   Protocol 6.4.5 (TTP.PostGadgetGen)**.** Given a secret key sk_{i-1}, do the following:

   (a) Let $g_i \leftarrow 0^{|g(sk_{i-1})|}$.

   (b) Populate $(\gamma_i^{\mathsf{in}}, \gamma_i^{\mathsf{mid}}, \gamma_i^{\mathsf{out}})$ with halves of EPR pairs. Send the other halves to VerDec via the side channel.

   Protocol 6.4.5 produces a "gadget" in which all qubits are replaced with halves of EPR pairs. These halves still get encrypted in Step 6c of Protocol 6.4.1. All other halves of these EPR pairs are sent to VerDec through the provided quantum side channel. $\mathsf{TTP}_2^{(\ell)}$.VerDec has access to the structural information g_ℓ (as this is sent via the classical side information channel from KeyGen to VerDec) and performs the necessary Bell measurements to recreate $\gamma_\ell^{\mathsf{in}}$, $\gamma_\ell^{\mathsf{mid}}$ and $\gamma_\ell^{\mathsf{out}}$ after the adversary has interacted with the EPR pair halves. Effectively, this postpones the generation of the gadget structure to decryption time. Of course, the measurement outcomes are taken into account by VerDec when calculating updates to the quantum one-time pad. As we will see in the description of $\mathsf{TTP}_4^{(\ell)}$, all corrections that follow the ℓth gadget are unaffected by the fact that the server cannot hold the correct information about these postponed measurements (not even in encrypted form).

3. $\mathsf{TTP}_3^{(\ell)}$ is the same as $\mathsf{TTP}_2^{(\ell)}$, except that gadget generation for the ℓth gadget is handled by TTP.FakeGadgetGen instead of TTP.PostGadgetGen (see Figure 6.8):

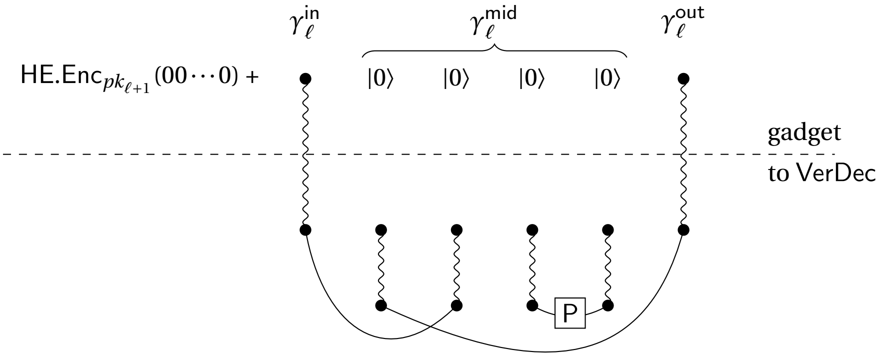

γ_ℓ^{in}
γ_ℓ^{mid}
γ_ℓ^{out}
$\mathsf{HE.Enc}_{pk_{\ell+1}}(00\cdots0)+$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
gadget
to VerDec
P

Figure 6.8: In $\mathsf{TTP}_3^{(\ell)}$, all of γ_ℓ^{mid} is replaced with dummy qubits. VerDec verifies the Bell measurements performed on these dummy qubits, and performs them on the top halves of the corresponding EPR pairs. Like in $\mathsf{TTP}_2^{(\ell)}$, VerDec also performs Bell measurements and a P gate on the lower halves.

Protocol 6.4.6 (TTP.FakeGadgetGen)**.** Given a secret key sk_{i-1}, do the following:

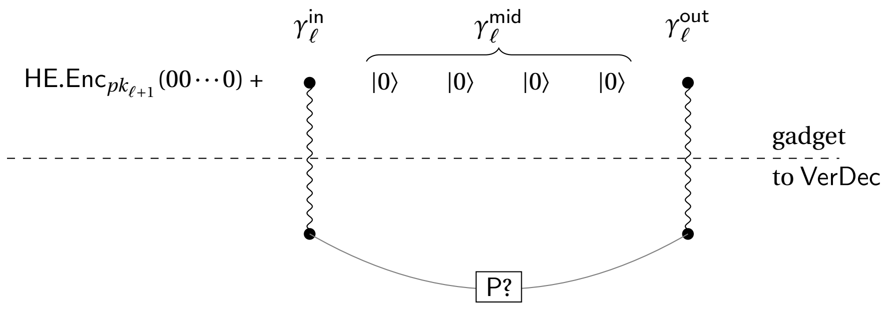
(a) Let $g_i \leftarrow 0^{|g(sk_{i-1})|}$.

(b) Populate $(\gamma_i^{\text{in}}, \gamma_i^{\text{mid}}, \gamma_i^{\text{out}})$ with halves of EPR pairs. Send the other halves to VerDec via the side channel.

(c) Send γ^{mid} to VerDec as well, returning the fake gadget with γ^{mid} replaced with zeros, as $(g_i, \gamma_i^{\text{in}}, |00\cdots0\rangle\langle 00\cdots0|\, \gamma_i^{\text{out}})$.

This algorithm prepares, instead of halves of EPR pairs, $|0\rangle$-states of the appropriate dimension for γ_ℓ^{mid}. (Recall that this dimension does not depend on $sk_{\ell-1}$). For γ_ℓ^{in} and γ_ℓ^{out}, halves of EPR pairs are still generated, as in $\mathsf{TTP}_2^{(\ell)}$. Via the side channel, the full EPR pairs for γ_ℓ^{mid} are sent to VerDec. As in the previous hybrids, the returned gadget is encrypted in TTP.KeyGen.

$\mathsf{TTP}_3^{(\ell)}$.VerDec verifies that the adversary performed the correct Bell measurements on the fake ℓth gadget by calling TC.VerDec. If this procedure accepts, $\mathsf{TTP}_3^{(\ell)}$.VerDec performs the same Bell measurements on the halves of the EPR pairs received from $\mathsf{TTP}_3^{(\ell)}$.KeyGen (and subsequently performs the Bell measurements that depend on g_ℓ on the other halves, as in $\mathsf{TTP}_2^{(\ell)}$). Effectively, $\mathsf{TTP}_3^{(\ell)}$.VerDec thereby performs the entire garden-hose protocol for HE.Dec on its own, removing the phase error in the process.



$\gamma_\ell^{\mathsf{in}}$
$\gamma_\ell^{\mathsf{mid}}$
$\gamma_\ell^{\mathsf{out}}$
$\mathsf{HE.Enc}_{pk_{\ell+1}}(00\cdots0)$ +
$|0\rangle$ $|0\rangle$ $|0\rangle$ $|0\rangle$
gadget
to VerDec
P?

Figure 6.9: In $\mathsf{TTP}_4^{(\ell)}$, the state that the evaluator receives is exactly equal to the state in $\mathsf{TTP}_3^{(\ell)}$ (see Figure 6.8). The only difference is the way VerDec applies the P gate (conditionally): instead of emulating the gadget usage, VerDec directly computes whether or not a phase needs to be applied, and performs the teleportation measurement on $\gamma_\ell^{\mathsf{in}}$ and $\gamma_\ell^{\mathsf{out}}$ accordingly.

4. $\mathsf{TTP}_4^{(\ell)}$ is the same as $\mathsf{TTP}_3^{(\ell)}$, except that VerDec (instead of performing the Bell measurements of the gadget protocol) uses its knowledge of the initial QOTP keys and all intermediate measurement outcomes to compute whether or not a phase correction is necessary after the ℓth T gate. $\mathsf{TTP}_4^{(\ell)}$.VerDec then performs this phase correction on the EPR half entangled with $\gamma_\ell^{\mathsf{in}}$, followed by a Bell measurement with the EPR half entangled with $\gamma_\ell^{\mathsf{out}}$. See Figure 6.9.

The first $\ell - 1$ gadgets in $\mathsf{TTP}_1^{(\ell)}$ through $\mathsf{TTP}_4^{(\ell)}$ are always functional gadgets, as in TTP. The last $t-\ell$ gadgets are all completely replaced by dummy states, and their functionality is completely outsourced to VerDec. In four steps, the functionality of the ℓth gadget is also transferred to VerDec. It is important to replace only one gadget at a time, because replacing a real gadget with a fake one breaks the functionality of the gadgets that occur later in the evaluation: the encrypted classical information held by the server does not correspond to the question of whether or not a phase correction is needed. By completely outsourcing the phase correction to VerDec, as is done for all gadgets after the ℓth one in all $\mathsf{TTP}_i^{(\ell)}$ schemes, we ensure that this incorrect classical information does not influence the outcome of the computation. Hence, correctness is maintained throughout the hybrid transformations.

We now show, in four separate lemmas, that these transformations of the scheme do not significantly affect the adversary's winning probability in the hybrid indistinguishability game.

Lemma 6.4.7. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all* $1 \leqslant \ell \leqslant t$,

$$\mathsf{AdvHyb}_{\mathsf{TTP}_1^{(\ell)}}^{\mathsf{TTP}_4^{(\ell+1)}}(\mathcal{A},\lambda) \leqslant \mathrm{negl}(\lambda). \tag{6.31}$$

Proof sketch. In $\mathsf{TTP}_4^{(\ell+1)}$, no information about $sk_{(\ell)}$ is sent to the adversary. In the original TTP scheme, the structure of the quantum state $\Gamma_{\ell+1}$ depended on it, but this structure has been replaced with dummy states in several steps in $\mathsf{TTP}_2^{\ell+1}$ through $\mathsf{TTP}_4^{\ell+1}$.

This is fortunate, since if absolutely no secret-key information is present, we are able to bound the difference in winning probability between $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_4^{(\ell+1)}}(\lambda)$ and $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_1^{\ell}}(\lambda)$ by reducing it to the q-IND-CPA security of the classical homomorphic-encryption scheme HE.

The proof is closely analogous to the proof of Lemma 6.3.4, and involves defining an adversary $\mathcal{A}'$ against the q-IND-CPA game that picks a random ℓ and simulates either $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_1^{(\ell)}}(\lambda)$ or $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_4^{(\ell+1)}}(\lambda)$ for a given adversary $\mathcal{A}$. Depending on whether that game is lost or won, $\mathcal{A}'$ will guess which of the two games she just simulated. Since the probability that she guesses correctly is negligibly close to $1/2$, there cannot (on average over ℓ) be a significant difference in winning probabilities between the two games.

Using the same derivation as in the proof of Lemma 6.3.4, we can conclude that for all $\mathcal{A}$ and all ℓ, the advantage of $\mathcal{A}$ in the game for $\mathsf{TTP}_1^{(\ell+1)}$ over $\mathsf{TTP}_4^{(\ell)}$ is negligible. This negligible function is independent of ℓ. □

Lemma 6.4.8. *For* $1 \leqslant \ell \leqslant t$ *and any QPT* $\mathcal{A}$*,* $\mathsf{AdvHyb}^{\mathsf{TTP}_2^{(\ell)}}_{\mathsf{TTP}_1^{(\ell)}}(\mathcal{A},\lambda) = 0$.

Proof. In $\mathsf{TTP}_1^{(\ell)}$, the ℓth error-correction gadget consists of a number of EPR pairs arranged in a certain order, as described by the garden-hose protocol for $\mathsf{HE.Dec}$. For example, this protocol may dictate that the ith and jth qubit of the gadget must form an EPR pair $|\Phi^+\rangle$ together. This can alternatively be achieved by creating two EPR pairs, placing half of each pair in the ith and jth position of the gadget state, and performing a Bell measurement on the other two halves. This creates a Bell pair $\mathsf{X}^a\mathsf{Z}^b|\Phi^+\rangle$ in positions i and j, where $a, b \in \{0,1\}$ describe the outcome of the Bell measurement.

From the point of view of the adversary, it does not matter whether these Bell measurements are performed during KeyGen, or whether the halves of EPR pairs are sent to VerDec for measurement. Because the key to the quantum one-time pad of the ℓth gadget is not sent to the adversary at all, the same state is created with a completely random Pauli in either case. Of course, the teleportation correction Paulis of the form $\mathsf{X}^a\mathsf{Z}^b$ need to be taken into account when updating the keys to the quantum one-time pad on the data qubits after the gadget is used. VerDec has all the necessary information to do this, because it observes the measurement outcomes, and computes the key updates itself (instead of decrypting the final keys from the computation log).

Thus, with the extra key update steps in $\mathsf{TTP}_2^{(\ell)}.\mathsf{VerDec}$, the inputs to the adversary are exactly the same in the games of $\mathsf{TTP}_1^{(\ell)}$ and $\mathsf{TTP}_2^{(\ell)}$. □

Lemma 6.4.9. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all* $1 \leqslant \ell \leqslant t$,

$$\mathsf{AdvHyb}^{\mathsf{TTP}_3^{(\ell)}}_{\mathsf{TTP}_2^{(\ell)}}(\mathcal{A},\lambda) \leqslant \mathrm{negl}(\lambda).$$

Proof. We show this bound by reducing the difference in winning probabilities between $\mathsf{TTP}_2^{(\ell)}$ and $\mathsf{TTP}_3^{(\ell)}$ to the IND-VER security of the somewhat-homomorphic scheme TC. Intuitively, because TC is IND-VER, if $\mathsf{TTP}_2^{(\ell)}$ accepts the adversary's claimed circuit of Bell measurements on the EPR pair halves, the effective map on those EPR pairs *is* the claimed circuit. Therefore, we might just as well ask VerDec to apply this map, as we do in $\mathsf{TTP}_3^{(\ell)}$, to get the same output state. If $\mathsf{TTP}_2^{(\ell)}$ rejects the adversary's claimed circuit on those EPR pair halves, then $\mathsf{TTP}_3^{(\ell)}$ should reject too. This is why we let the adversary act on an encrypted dummy state of $|0\rangle$s.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be a set of QPT algorithms on the appropriate registers, so that we can consider it as an adversary for the hybrid indistinguishability game for either $\mathsf{TTP}_2^{(\ell)}$ or $\mathsf{TTP}_3^{(\ell)}$, for any ℓ (see Game 5.6.8 and Figure 5.6). The input/output wires to the adversary in both these games are identical, so we can evaluate $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda) = 1]$ and $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda) = 1]$ for the same $\mathcal{A}$.

Now define an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3)$ for the IND-VER game against TC, $\mathsf{VerGame}_{\mathcal{A}',\mathsf{TC}}(\lambda)$, as follows:

$\mathcal{A}'_1$ samples $\ell \in_R [t]$. She runs $\mathsf{TTP}_2^{(\ell)}.\mathsf{KeyGen}$ until right before she is supposed to call $\mathsf{TTP.GadgetGen}(sk_{\ell-1})$, in the ℓth iteration of that loop. Up to this point, $\mathsf{TTP}_2^{(\ell)}.\mathsf{KeyGen}$ is identical to $\mathsf{TTP}_3^{(\ell)}.\mathsf{KeyGen}$. It has generated real gadgets Γ_1 through $\Gamma_{\ell-1}$, and has not yet used the freshly sampled permutation π_ℓ. $\mathcal{A}'_1$ now sets g_ℓ to be the all-zero string, and generates halves of EPR pairs for $\gamma_\ell^{\mathsf{in}}$, $\gamma_\ell^{\mathsf{mid}}$ and $\gamma_\ell^{\mathsf{out}}$. She sends $\gamma_\ell^{\mathsf{mid}}$ to the challenger via the register X, and everything else (including sk) to $\mathcal{A}'_2$ via the side register R.

$\mathcal{A}'_2$ continues $\mathsf{TTP}_2^{(\ell)}.\mathsf{KeyGen}$, using the response from the challenger instead of encrypting $\gamma_\ell^{\mathsf{mid}}$ herself. Again, this part of the key-generation procedure is identical for $\mathsf{TTP}_2^{(\ell)}$ and $\mathsf{TTP}_3^{(\ell)}$. Call the resulting evaluation key ρ_{evk}. $\mathcal{A}'_2$ starts playing the hybrid indistinguishability game with $\mathcal{A}$, as follows:

1. Flip a bit $r \in \{0,1\}$.
2. Send ρ_{evk} to $\mathcal{A}_1$. If $r = 0$, encrypt the response of $\mathcal{A}_1$ using the secret key sk generated by $\mathcal{A}'_1$. If $r = 1$, encrypt a $|0\rangle$ state of appropriate dimension instead. Note that for this encryption, the permutation π_ℓ is also not needed.
3. Send the resulting encryption, along with the side info from $\mathcal{A}_1$, to $\mathcal{A}_2$.

4. On the output of $\mathcal{A}_2$, start running $\mathsf{TTP}_2^{(\ell)}.\mathsf{VerDec}$ until the actions on the ℓth gadget need to be verified. Since the permutation on the state $\gamma_\ell^{\mathsf{mid}}$ is unknown to $\mathcal{A}_2'$ (it was sent to the challenger for encryption), she cannot verify this part of the computation.
5. Instead, send the relevant part of the computation log to the challenger for verification, along with the relevant part of the claimed circuit (the Bell measurements on the gadget state), and the relevant qubits, all received from $\mathcal{A}_2$.
6. In the meantime, send the rest of the working memory to $\mathcal{A}_3'$ via register R'.

$\mathcal{A}_3'$ continues the simulation of the hybrid game with $\mathcal{A}$, as follows:

1. If the challenger rejects, output $|\bot\rangle\langle\bot| \otimes |\mathsf{rej}\rangle\langle\mathsf{rej}|$.
2. If the challenger accepts, then we know that the challenger applies the claimed subcircuit to the quantum state it did not encrypt (either $|0\rangle$ or $\gamma_\ell^{\mathsf{mid}}$, depending on the bit the challenger flipped), and possibly swaps this state back in (again depending on which bit it flipped). Continue the $\mathsf{TTP}_2^{(\ell)}.\mathsf{VerDec}$ computation for the rest of the computation log.
3. Send the result (the output quantum state, the claimed circuit, and the accept/reject flag) to $\mathcal{A}_3$, and call its output bit r'.

The adversary $\mathcal{A}_3'$ then outputs 0 if $r = r'$, and 1 otherwise.

Recall from Game 5.6.8 (see Figure 5.6) that the challenger flips a coin (let us call the outcome $s \in \{0,1\}$) to decide whether to encrypt the quantum state provided by $\mathcal{A}'$, or to swap in an all-zero dummy state before encrypting. Keeping this in mind while inspecting the definition of $\mathcal{A}'$, one can see that whenever $s = 0$, $\mathcal{A}'$ takes the role of challenger in the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda)$ with $\mathcal{A}$, and whenever $s = 1$, she plays $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda)$. Now let us consider when the newly defined adversary $\mathcal{A}'$ wins the VER indistinguishability game for TC. If $s = 0$, $\mathcal{A}'$ needs to output a bit $s' = 0$ to win. This happens, by definition of $\mathcal{A}'$, if and only if $\mathcal{A}$ wins the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda)$ (i.e., $r = r'$). On the other hand, if $s = 1$, $\mathcal{A}'$ needs to output a bit $s' = 1$ to win. This happens, by definition of $\mathcal{A}'$, if and only if $\mathcal{A}$ loses the game $\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda)$ (i.e., $r \neq r'$). Thus the winning probability of $\mathcal{A}'$ is:

$$\Pr[\mathsf{VerGame}_{\mathcal{A}',\mathsf{TC}}(\lambda) = 1] \tag{6.32}$$

$$= \mathop{\mathbb{E}}_{\ell}\Big(\Pr[s=0]\cdot\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda)=1] + \Pr[s=1]\cdot\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda)=0]\Big) \tag{6.33}$$

$$= \mathop{\mathbb{E}}_{\ell}\left(\frac{1}{2}\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda)=1] + \frac{1}{2}\Big(1 - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda)=1]\Big)\right) \tag{6.34}$$

$$= \frac{1}{2} + \frac{1}{2} \mathop{\mathbb{E}}_{\ell} \Big(\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_2^{(\ell)}}(\lambda) = 1] - \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}_3^{(\ell)}}(\lambda) = 1] \Big). \tag{6.35}$$

From the IND-VER property of TC (see Lemma 5.5.6) we know that Equation (6.32) is at most $1/2 + \mathrm{negl}(\lambda)$. From this, and the fact that t is polynomial in λ, the statement of the lemma follows. □

Lemma 6.4.10. *For any QPT $\mathcal{A}$, there exists a negligible function* negl *such that for all* $1 \leqslant \ell \leqslant t$,

$$\mathsf{AdvHyb}_{\mathsf{TTP}_3^{(\ell)}}^{\mathsf{TTP}_4^{(\ell)}}(\mathcal{A}, \lambda) \leqslant \mathrm{negl}(\lambda).$$

Proof. Let $f(s)$ be the bit that, after the ℓth T gate, determines whether or not a phase correction is necessary. Here, s is all the relevant starting information (such as quantum-one-time-pad keys, gadget structure, permutations, measurement outcomes, and applied circuit), and f is some function that determines the X key on the relevant qubit right before application of the T gate.

In $\mathsf{TTP}_3^{(\ell)}$, a phase correction after the ℓth T gate is applied conditioned on the outcome of

$$\mathsf{HE.Dec}_{sk_{\ell-1}}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_{\ell-1}}(\mathsf{HE.Enc}_{pk_0}(s))), \tag{6.36}$$

because the garden-hose computation in the gadget computes the classical decryption. In the above expression, we again slightly abuse notation: as in the proof of Lemma 6.4.4, we include recryption steps in $\mathsf{HE.Eval}_{evk_0,\ldots,evk_{\ell-1}}$. As long as t is polynomial in λ, we have for all $\ell \leqslant t$, by correctness of HE,

$$\Pr\Big[\mathsf{HE.Dec}_{sk_{\ell-1}}(\mathsf{HE.Eval}^{f}_{evk_0,\ldots,evk_{\ell-1}}(\mathsf{HE.Enc}_{pk_0}(s))) \neq f(s)\Big] \leqslant \mathrm{negl}(\lambda). \tag{6.37}$$

In $\mathsf{TTP}_4^{(\ell)}$, the only difference from $\mathsf{TTP}_3^{(\ell)}$ is that, instead of performing the garden-hose computation on the result of the classical homomorphic-evaluation procedure, the phase correction is applied directly by VerDec, conditioned on $f(s)$. The probability that in $\mathsf{TTP}_4^{(\ell)}$, a phase is applied (or not) when in $\mathsf{TTP}_3^{(\ell)}$ it is not (or is), is negligible. □

Final hybrid: Removing all classical FHE

In $\mathsf{TTP}_4^{(1)}$, all of the error-correction gadgets have been removed from the evaluation key, and the error-correction functionality has been redirected to VerDec completely. Effectively, $\mathsf{TTP}_4^{(1)}.\mathsf{KeyGen}$ samples a permutation π, generates a lot of magic states (for P, H and T) and encrypts them using $\mathsf{TC.Enc}_\pi$, after which the keys to the quantum one-time pad used in that encryption are homomorphically encrypted under pk_0. The adversary receives those encrypted values, and uses them in its homomorphic classical computation. However, since the computation contains fake gadgets, the homomorphically computed output one-time-pad keys will not be correct. VerDec uses the information sent via its side channels to compute the correct keys (but still checks the log for syntactical consistency).

The final step is to define a hybrid TTP^*, similarly to TCL^*, where all classical information is replaced with zeros before encrypting. The adversary is allowed to act on those encryptions, but while its homomorphic computations are syntactically checked in the log, VerDec does not decrypt and use the resulting values. This allows us to link $\mathsf{TTP}_4^{(1)}$ to a final hybrid, TTP^*, where all classical information is replaced with zeros before encrypting.

Lemma 6.4.11. *For any QPT $\mathcal{A}$,* $\mathsf{AdvHyb}^{\mathsf{TTP}^*}_{\mathsf{TTP}_4^{(1)}}(\mathcal{A},\lambda) \leqslant \text{negl}(\lambda)$.

Proof. Identical to the proof of Lemma 5.6.11: by reduction to q-IND-CPA security of the classical scheme HE. □

Proof of main theorem

Just like TCL^*, the scheme TTP^* is very similar to TC. This similarity allows us to bound the success probability of any adversary in the hybrid indistinguishability game for TTP^*.

Considering TTP^* in more detail, we can see that it is actually very similar to TC. This allows us to prove the following lemma, which is the last ingredient for the proof of verifiability of TTP.

Lemma 6.4.12. *For any QPT $\mathcal{A}$,* $\Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}^*}(\lambda) = 1] \leqslant 1/2 + \text{negl}(\lambda)$.

Proof. Identical to the proof of Lemma 5.6.12: by observing the similarities between TTP^* and TC (the presence of EPR pair halves does not affect the correspondence between these two schemes), and using the fact that TC is IND-VER-2 (Definition 5.5.4). □

Putting together Lemmas 6.4.3, 6.4.4 and 6.4.7 to 6.4.12, we are able to prove the main result of this section:

Theorem 6.4.13. *The vQFHE scheme* TTP *satisfies SEM-VER.*

Proof. From Lemmas 6.4.3, 6.4.4 and 6.4.7 to 6.4.11, we may conclude that if t (the number of T gates in the circuit) is polynomial in λ (the security parameter), then for any polynomial-time adversary $\mathcal{A}$,

$$\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TTP}}(\lambda) = 1] \;-\; \Pr[\mathsf{Hyb}_{\mathcal{A},\mathsf{TTP}^*}(\lambda) = 1] \;\leqslant\; \text{negl}(\lambda), \tag{6.38}$$

since the sum of polynomially many negligible terms is negligible (it is important to note that there is only a constant number of *different* negligible terms involved). By Lemma 6.4.12, which reduces verifiability of TTP^* to verifiability of TC, it follows that $\Pr[\mathsf{VerGame}_{\mathcal{A},\mathsf{TTP}}(\lambda) = 1] \leqslant 1/2 + \text{negl}(\lambda)$, i.e., that TTP is IND-VER. By Lemma 5.4.8, TTP is also SEM-VER. □

6.5 Application: quantum one-time programs

We briefly sketch an application of a vQFHE scheme like TTP to one-time programs. A classical one-time program (or cOTP[2] is an idealized object which can be used to execute a function once, but then self-destructs. In the case of a quantum OTP (or qOTP), the program executes a quantum channel Φ. In the usual formalization, Φ has two inputs and is public. One party (the sender) creates the qOTP by fixing one input, and the qOTP is executed by a receiver who selects the other input. To recover the intuitive notion of one-time programs, choose Φ to be a universal circuit. Following the approach of Broadbent et al. [BGS13], we thus consider the ideal functionality of a qOTP.

Definition 6.5.1 (Ideal quantum one-time programs [BGS13, Functionality 3])**.** The ideal functionality $\mathcal{F}_{\Phi}^{\mathsf{OTP}}$ for a channel $\Phi^{XY \to Z}$ is the following:

1. ***Create:*** given register X from the sender, store X and send create to the receiver.
2. ***Execute:*** given register Y from the receiver, return Φ applied to XY to the receiver. Delete any trace of its execution.

As in the work of Broadbent et al. [BGS13], we only allow corrupting receivers; unlike that work, we consider computational (rather than statistical) security. The achieved result is therefore slightly weaker. The construction within our vQFHE framework is however much simpler, and shows how applications can be constructed using vQFHE as a black box.

The construction

Choose a vQFHE scheme $S = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{VerDec})$ satisfying SEM-VER. For simplicity, we first describe the classical input/output case, i.e., the circuit begins and ends with classical registers only. Let c be such a circuit, for the map $\Phi^{XY \to Z}$. On ***Create***, the sender generates keys $(sk, \rho_{evk}) \leftarrow \mathsf{KeyGen}$ and encrypts his input σ^X using sk. The sender also generates a cOTP for the public, classical function VerDec, choosing the circuit and key inputs to be c and sk; the input field for the computation log is left open for the receiver to select. The qOTP is then the triple

$$\Xi_c := \left(\rho_{evk}, \mathsf{Enc}_{sk}(\sigma^X), \mathsf{cOTP}_{\mathsf{VerDec}}(c, sk, \cdot)\right).$$

On ***Execute***, the receiver computes as follows. The receiver's (classical) input Y together with the (public) circuit c defines a homomorphic computation on the ciphertext $\mathsf{Enc}_k(\sigma^X)$, which the receiver can perform using Eval and ρ_{evk}. Since c has only

[2]The abbreviations "cOTP" and "qOTP" in this section should not be confused with the abbreviation (Q)OTP for (quantum) one-time pad.

classical outputs, the receiver measures the final state completely. At the end of that computation, the receiver holds the (completely classical) output of the computation log from Eval. The receiver plugs the log into $\mathsf{OTP}_{\mathsf{VerDec}}(c, sk, \cdot)$, which produces the decrypted output.

We handle the case of arbitrary circuits c (with quantum input and output) as follows. Following the ideas of Broadbent et al. [BGS13], we augment the above qOTP with two auxiliary quantum states: an "encrypt-through-teleport" gadget $\sigma_{\mathsf{in}}^{V_1V_2}$, and a "decrypt-through-teleport" gadget $\sigma_{\mathsf{out}}^{W_1W_2}$. These are maximally entangled states with the appropriate map (encrypt or decrypt) applied to one half (V_2 or W_2). The receiver uses teleportation through the V_1 register to encrypt their input register Y before evaluating, and places the teleportation measurements into the computation log. After evaluation, the receiver uses the W_1 register to teleport the ciphertext out, combining the teleportation measurements with the output of $\mathsf{cOTP}_{\mathsf{VerDec}}(c, sk)$ to compute the final quantum-one-time-pad decryption keys.

Security sketch

Starting with a QPT adversarial receiver $\mathcal{A}$ which attacks the real functionality (that contains an ideal cOTP), we construct a QPT simulator $\mathcal{S}$ which attacks the ideal functionality with similar success probability. We split $\mathcal{A}$ into $\mathcal{A}_1$ (receive input, output the cOTP query and side information) and $\mathcal{A}_2$ (receive result of cOTP query and side information, produce final output). The simulator $\mathcal{S}$ will generate its own keys, and provide fake gadgets that will trick $\mathcal{A}$ into teleporting its input to $\mathcal{S}$, who will then use that input on the ideal functionality.

The simulator, upon receiving the "create" message, generates $(sk, \rho_{evk}) \leftarrow \mathsf{KeyGen}$ and encrypts a dummy input $|0\rangle\langle 0|^X$ via Enc_{sk}. Instead of the encryption gadget $\sigma_{\mathsf{in}}^{V_1V_2}$, $\mathcal{S}$ provides half of a maximally entangled state in register V_1 and likewise in register V_2. The other halves V_1' and V_2' of these entangled states are kept by $\mathcal{S}$. The same is done in place of the decryption gadget $\sigma_{\mathsf{out}}^{W_1W_2}$, with $\mathcal{S}$ keeping W_1' and W_2'. Then $\mathcal{S}$ runs $\mathcal{A}_1$ with input $\rho_{evk}, \mathsf{Enc}_{sk}(|0\rangle\langle 0|^X)$ and registers $YV_1V_2W_1W_2$. It then executes VerDec_{sk} on the output (i.e., the cOTP query) of $\mathcal{A}_1$ to see if she correctly followed the Eval protocol. If she did not, then $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ plugs register V_1' into the ideal functionality, and teleports the output into register W_2. Before responding to $\mathcal{A}_2$, it corrects the one-time pad keys appropriately using its teleportation measurements.

6.6 Practical considerations

In a setting where a less powerful client wants to delegate some quantum computation to a more powerful server, it is important to minimize the amount of effort required from the client. In delegated quantum computation, the complexity of a protocol can be measured by, among other things, the total amount of communication between client and server, the number of rounds of communication, and the quantum resources available to the client, such as possible quantum operations and memory size.

The constructions in this chapter rely heavily on classical homomorphic encryption. Although implementations of homomorphic encryption are starting to appear [DS15; CLP17; DM17; HS20], they still tend to have a fairly large computational overhead. In addition to that, our quantum homomorphic-encryption schemes TP and TTP require the creation of polynomial-size gadget states, for which there is a priori no bound on the polynomial.

In this section, we describe some considerations on the efficiency of the schemes TP and TTP. In Section 6.6.1, we describe how to construct the error-correcting gadgets for a specific classical homomorphic-encryption scheme. In Section 6.6.2, we describe how a client can generate the error-correcting gadgets for TP using only the SWAP operation and Paulis, if he has an (untrusted) source of EPR pairs.

6.6.1 Gadget construction from LWE

The classical FHE scheme by Brakerski and Vaikuntanathan [BV14] is well-suited for our construction, and its decryption function is representative for a much wider class of schemes which are based on the hardness of the learning-with-errors problem. Based on its decryption function, we construct gadgets for TP and TTP. Of course, we already know how to construct the gadgets using the garden-hose complexity of the decryption function (Theorem 6.2.1), but in this section we give an explicit, potentially more efficient construction.

Let λ be the security parameter, and let p be the modulus of the integer ring over which the scheme from Brakerski and Vaikuntanathan [BV14] operates.

The ciphertext c is given by a pair $(\mathbf{v}, w)$, with $\mathbf{v} \in \mathbb{Z}_p^\lambda$ and $w \in \mathbb{Z}_p$. The secret key $\mathbf{s}$ is an element of $\mathbb{Z}_p^k$. The decryption into a plaintext m involves computation of an inner product over the ring $\mathbb{Z}_p$,

$$m = (w - \langle \mathbf{v}, \mathbf{s} \rangle) \pmod{p} \pmod{2}. \tag{6.39}$$

Brakerski and Vaikuntanathan are able to ensure that the modulus p is small, i.e., polynomial in λ, before encryption. Below, we present an explicit construction for this case of small modulus p. In case the modulus is superpolynomially large, constructing the gadget explicitly appears to be much harder, but we can still apply Theorem 6.2.1 to convert the decryption circuit, which has depth $O(\log\lambda + \log\log p)$ [BV14, Lemma 4.5], to obtain a polynomial-size gadget. In that case, we do not exploit the specific structure of the decryption function to construct a more efficient gadget.

For the rest of this section, take the modulus p to be polynomial in λ. We describe a series of small "permutation gadgets" that move an arbitrary qubit to a location,

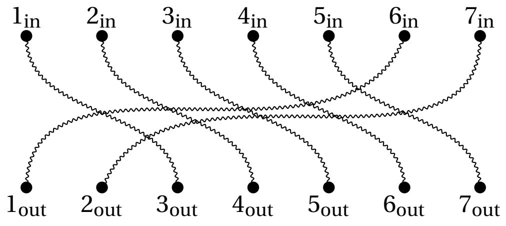


Figure 6.10: A single permutation gadget for $p = 7$ and $q = 2$. This gadget effectively adds 2 modulo 7 to the position of the input qubit. All $2p = 14$ qubits in the gadget are given to the evaluator. She connects it to other subgadgets, in a way dictated by the values in $\mathbf{v}$, by performing Bell measurements between each i_{out} of the depicted gadget and i_{in} of the next gadget. If the depicted gadget is connected to another subgadget with $p = 7$ and, e.g., $q = 1$, the result is a permutation gadget for $p = 7$ and $q = 3$.

depending on whether $m = 0$ or $m = 1$. By appending the reverse of this construction as in Section 6.2.3, one can turn these into a gadget which applies an inverse phase gate whenever $m = 1$.

We rewrite Equation (6.39) in terms of binary arithmetic [BV14, Section 4.5]. Let $\mathbf{s}[i][j]$ denote the jth bit of the ith entry of $\mathbf{s}$, so that the inner product can be written as

$$\begin{aligned} w - \langle \mathbf{v}, \mathbf{s} \rangle \pmod p &= w - \sum_{i=1}^{\lambda} \mathbf{v}[i]\mathbf{s}[i] \pmod p \\ &= w - \sum_{i=1}^{\lambda} \sum_{j=0}^{\log_2 p} \mathbf{v}[i][j] \cdot 2^j \cdot \mathbf{s}[i] \pmod p \qquad (6.40) \end{aligned}$$

A *permutation gadget* is now a subgadget of size $2p$, parametrized by a number $q \in \mathbb{Z}_p$. Label the first p qubits by 0_{in} to $(p-1)_{\text{in}}$, and the second p qubits by 0_{out} to $(p-1)_{\text{out}}$. The gadget simply creates EPR pairs between x_{in} and $(x + q \pmod p))_{\text{out}}$, for all $x \in \mathbb{Z}_p$. Such a gadget can effectively simulate addition with q over $\mathbb{Z}_p$. See Figure 6.10.

For each i from 1 to λ, and each j from 0 to $\log_2 p$, we create a permutation gadget, labeled by (i, j), for the number $2^j \cdot \mathbf{s}[i]$, representing the $\lambda \log_2 p$ terms in Equation (6.40). Each of the terms in that sum contributes to the total, or not, depending on the bit $\mathbf{v}[i][j]$ of the ciphertext.

The evaluator uses the collection of permutation gadgets in the following way. She performs a Bell measurement between the input qubit and the 0_{in} qubit of the first gadget such that $\mathbf{v}[i][j] = 1$. Then, she connects all output qubits (0_{out} through $(p-1)_{\text{out}}$) of this gadget to all the input qubits of the next gadget for which $\mathbf{v}[i][j] = 1$.

After teleporting her qubit through all gadgets, the qubit will be exactly at the location z_{out} of the final gadget the evaluator used, where $z_{\text{out}} = \sum_{i=1}^{\lambda} \mathbf{v}[i]\mathbf{s}[i] \pmod p$.

The evaluator does not know at which of the p locations the qubit is, but she can apply an inverse phase gate to those positions z for which $w - z = 1 \pmod 2$.

Finally, we reverse the entire construction to route the unknown qubit back to a known location, as in Section 6.2.3. The size of the total gadget is then bounded by $4\lambda p \log_2 p$.

6.6.2 Constructing gadgets using limited quantum resources

The nonverifiable scheme TP gives rise to a three-round delegated quantum computation protocol in a setting where the client can perform only Pauli and SWAP operations. TP.Enc and TP.Dec already only require local application of Pauli operators to a quantum state, but TP.KeyGen is more involved because of the gadget construction. However, when supplied with a set of EPR pairs from the server (or any other untrusted source), the client can generate the quantum evaluation key for TP using only Pauli and swap operations. Even if the server produces some other state than the claimed list of EPR pairs, the client can prevent the leakage of information about her input by encrypting the input with random Pauli operations.

The gadget $\Gamma_{pk'}(sk)$ is essentially a list of $2m$ random Bell pairs (some of which have an extra $\mathsf{P}^\dagger \otimes \mathsf{I}$ applied to them), where the qubits are ordered in some way that depends on *sk*. If the key generator is supplied with a list of $2m$ EPR pairs $|\Phi^+\rangle$, and as many pairs $(\mathsf{I} \otimes \mathsf{P}^\dagger)|\Phi^+\rangle$, he can create the gadget by swapping some of the qubits, and applying random Pauli operations (using X and Z gates) on every pair. Any unused pairs are discarded.

If the (potentially malicious) supplier of these pairs follows the protocol and sends actual EPR pairs to the key generator, the above procedure suffices to hide all information about *sk*. However, if the supplier acts maliciously, he may send two qubits to the key generator claiming that they form an EPR pair, while in reality he is keeping some form of entanglement with one or both of the qubits. We need to make sure that even in this case, where the supplier actively tries to gather information about *sk*, this information remains private.

The key generator, upon receiving the (real or fake) EPR pairs, can apply independently selected random Pauli transformations on every qubit. If the qubits really formed EPR pairs, it would suffice to apply a random Pauli to only one of the two qubits in the pair, but by applying this transformation to both qubits, any entanglement that a malicious supplier might hold with any of them becomes completely useless. Since any swap of two qubits consists of three CNOT gates that commute with the Pauli's, the state after swapping the qubits into the correct order is still completely mixed. Hence, no information about *sk* is revealed to the supplier.

For the scheme TTP, although the above strategy suffices to ensure privacy, it does not guarantee that the outcome of the evaluation will be correct. The scheme TTP loses its advantage over TP, especially since we would have to ask the untrusted supplier to supply CSS-encoded EPR pairs, *and* CSS-encoded magic states.

Alternatively, TP can be regarded as a two-round delegated-quantum-computation protocol in a setting where the client can perform arbitrary Clifford operations, but is limited to a constant-sized quantum memory, given that HE.Dec is in NC^1. In that case, Barrington's theorem [Bar89] tells us that the measurements in the garden-hose model are fairly "local" (i.e., do not connect pipes that are very far away), and that the gadgets can be constructed ten qubits at a time [DSS16]. By decomposing the permutations in the construction of Barrington's theorem into 2-cycles, the quantum memory can even be reduced to only four qubits. The client sends these small parts of the gadgets to the server as they are completed. Because communication remains one-way until all gadgets have been sent, this staggered sending can be regarded as a single round of communication.

The verifiable scheme TTP can also be reduced to a constant-size version by combining the above gadget-generation strategy with the constant-memory trap code (see Section 3.3.3). The IND-VER security error is then reduced from negligible to inverse polynomial, but the privacy error remains negligible.

6.7 Conclusion

This chapter was centered around the "T-gate gadget", a quantum state that aids in the homomorphic evaluation of a T gate by removing a potential phase error from the ciphertext. The T-gate gadgets appeared in both the nonverifiable QFHE scheme TP, and the vQFHE scheme TTP. The gadget construction is based on a new and interesting connection between the area of instantaneous nonlocal quantum computation and quantum homomorphic encryption. Speelman's techniques, based on the garden-hose model, have turned out to be crucial for our construction.

The structure of the evaluation key is fairly modular, consisting of exactly one gadget for every T gate. The evaluation of a T gate does not cause errors to accumulate on the quantum state. TP is very compact in the sense that the state of the system after the evaluation of a T gate has the same form as after the initial encryption, except for any classical changes caused by the classical FHE evaluation. This kind of compactness also implies that individual evaluation gadgets can be supplied "on demand" by the holder of the secret key. Once an evaluator runs out of gadgets, the secret key holder can simply supply more of them.

TP is the first quantum homomorphic-encryption scheme that is compact and allows evaluation of circuits with polynomially many T gates in the security parameter, i.e., arbitrary polynomial-size circuits. Assuming that the number of wires involved in the evaluation circuit is also polynomially related to the security parameter, we may consider TP to be leveled fully homomorphic. The scheme is based on an arbitrary classical FHE scheme, and any computational assumptions needed for the classical scheme are also required for security of TP. However, since TP uses the classical

FHE scheme as a black box, any FHE scheme can be plugged in to change the set of computational assumptions.

We also presented a new quantum-cryptographic primitive: quantum fully-homomorphic encryption with verification (vQFHE). Using the trap code for quantum authentication [BGS13] and the garden-hose gadgets, we constructed a vQFHE scheme TTP which satisfies (1) correctness, (2) compactness, (3) secure verifiability, (4) q-IND-CPA privacy, and (5) authentication.

In terms of applications, TP can be appreciated as a constant-round scheme for *blind delegated quantum computation*, using computational assumptions. The server can evaluate a universal quantum circuit on the encrypted input, consisting of the client's quantum input and a (classical) description of the client's circuit. In this context, it is desirable to minimize the quantum resources needed by the client. We argued that our scheme can still be used for constant-round blind delegated quantum computation if we limit either the client's quantum memory or the types of quantum operations the client can perform.

As another application, we can instantiate TP with a classical *multi-key* FHE scheme that allows for multiple clients to encrypt (and later jointly decrypt) their inputs to a joint computation. In a paper subsequent to this work, Goyal details how this can be done, and how the multi-key QFHE can provide multi-party quantum computation in certain settings [Goy18].

Adding verification immediately yields new applications of QFHE, e.g., allowing users of a "quantum cloud service" to certify the server's computations. Furthermore, we showed that verifiable QFHE leads to a simple construction of quantum one-time programs (qOTPs). In this construction, the qOTP for a functionality Φ_c consists of an evaluation key and a classical one-time program which performs vQFHE verification for Φ_c only.

6.7.1 Future directions

We leave open several interesting directions for applications of (v)QFHE. Classically, FHE has many applications [ABF+13; BR14], and defining and constructing their appropriate analogues would be a valuable addition to the field of quantum cryptography. Multi-party computation has already been constructed form QFHE in a limited security setting [Goy18], and can potentially be generalized. We also consider it likely that our new techniques will be useful in other contexts such as quantum indistinguishability obfuscation [AF16].

Since the construction of TP described in Section 6.3 first appeared online [DSS16], the required computational power has been greatly reduced: Mahadev showed that nonverifiable quantum fully homomorphic encryption is possible with a purely classical client [Mah18a]. The required computational assumptions were later relaxed to a more standard version of learning with errors [Bra18]. Mahadev's protocol, and its main ingredient of trapdoor claw-free function families, has proven useful in many related applications [BCM+18; BKVV20]. In a context where verification is not required, it is preferable to use these new classical-client schemes [Mah18a; Bra18], as opposed to the TP scheme. In fact, in Chapter 7, we will do exactly that.

In the context of verification, however, the scheme TTP presented in this chapter is currently still the only known homomorphic one. Significant progress has been made in the domain of verification of quantum computation (see also the discussion in Section 5.1), but unresolved questions remain: it is still unknown whether verifiable quantum homomorphic encryption is possible with a classical client, or whether there exist vQFHE schemes where verification can be done publicly (i.e., without the secret decryption key).

7 Impossibility of quantum obfuscation

Chapter contents

7.1 Introduction

The obfuscation of a circuit is an object, typically another circuit, that allows a user to evaluate the functionality of the original circuit without learning any additional information about the structure of the circuit. Obfuscation is useful for publishing software without revealing the code, but it also has more fundamental applications in cryptography. For example, the strongest notion called *virtual black-box* obfuscation can transform any private-key encryption scheme into a public-key scheme, and transform public-key schemes into fully-homomorphic schemes. Unfortunately, this strongest notion turns out to be impossible for general circuits [BGI+01] – at least, if we require the obfuscation of a circuit to be a circuit itself.

The classical impossibility result leaves open an intriguing possibility: what if the obfuscation of a (classical) circuit is allowed to be a *quantum state*? Could a quantum state capture all the information about a functionality, allowing a user to produce correct outputs, without revealing all that information? This possibility seems hopeful, due to the unrevealing nature of quantum states. However, in this work, we show that virtual-black-box obfuscating classical circuits into quantum states is not possible.

Barak et al. defined the obfuscating property of virtual black-box (vbb) obfuscators as follows: any information that an adversary can learn about a circuit from its obfuscation can also be learned by a simulator that does not have access to the obfuscation, but only to an oracle for the circuit's functionality [BGI+01]. In this definition, the crucial difference between the adversary and the simulator is that the adversary has access to a short representation of the circuit (namely, the obfuscation), whereas the simulator only has access to an input/output interface that implements the functionality. Some circuit classes allow the adversary to exploit this difference by using the obfuscation as an input value to the circuit itself. Those circuit classes are unobfuscatable in the vbb sense, rendering vbb obfuscation impossible for the general class of circuits in **P** [BGI+01].

In more detail, the impossibility proof from Barak et al. [BGI+01] relies on point functions, which output zero everywhere except at a single input value α, where they output a string β. The circuits in the unobfuscatable class can, depending on the input, do all of the following: (1) apply that point function, (2) return an encryption of α, (3) homomorphically evaluate a gate, or (4) check whether a ciphertext decrypts to β. An adversary holding the obfuscation is able to divide it into single gates, and can use those to homomorphically evaluate option (1), thereby converting a ciphertext for α into a ciphertext for β. That way, the adversary can tell whether she is holding an obfuscation with a point function from α to β, or one with the all-zero function. (In the second case, the homomorphic evaluation would yield a ciphertext for zero, rather than one for β. This distinction can be tested using option (4).) A simulator, only having access to the input/output behavior, cannot perform the homomorphic evaluation, because it cannot divide the functionality into single gates.

The above construction rules out the existence of an obfuscator that maps classical circuits to classical circuits. It leaves open the possibility of an obfuscator that maps classical circuits to *quantum states*: such a quantum state, together with a fixed public "interpreter map", could be used to evaluate the obfuscated circuit. The possibility of quantum obfuscation was the object of study for Alagic and Fefferman [AF16], who attempted to port the classical impossibility proof to the quantum setting. In doing so, they encountered two issues:

Homomorphic evaluation. The interpreter map, that runs the obfuscation state on a chosen input, is a quantum map. It will likely have quantum states as intermediate states of the computation, so in order to homomorphically run the point function, one needs the ability to evaluate quantum gates on quantum ciphertexts. Functionality (3) described above would become a quantum functionality, and the unobfuscatable circuit class will thus need to contain quantum circuits to perform homomorphic evaluation steps.

Reusability. In the construction by Barak et al. [BGI+01], the obfuscated circuit needs to be used multiple times: for example, each homomorphic gate evaluation requires a separate call to the obfuscated circuit. If the obfuscation is a (classical or quantum) circuit, this poses no problem, but if it is a quantum *state*, multiple uses are not guaranteed.

These two issues limit the extent of the impossibility results in the quantum setting: it is only known to be impossible to vbb obfuscate *quantum* circuits into *reusable* obfuscated states (e.g., quantum circuits) [AF16].

After it became clear [BGI+01] that obfuscating all classical circuits is impossible, efforts were made to construct obfuscators for smaller, but still nontrivial, classes of circuits. Successful constructions have been found for several classes of evasive functions, such as point functions [Wee05; CD08] and compute-and-compare functions [WZ17; GKW17]. Currently, no quantum obfuscators are known for circuit classes that cannot be classically obfuscated.

7.1.1 Contributions

We strengthen the impossibility of virtual-black-box obfuscation of classical circuits by showing that classical circuits cannot be obfuscated into quantum states. We assume the existence of classical-client quantum fully homomorphic encryption and classical obfuscation of compute-and-compare functions. Both of these can be constructed from the learning-with-errors (LWE) assumption [Mah18a; Bra18; WZ17; GKW17]. The compute-and-compare construction requires the strongest assumption in terms of the LWE parameters.

Theorem (informal). *If LWE is hard for quantum algorithms, then it is impossible to quantum vbb obfuscate the class of polynomial-size classical circuits (even with* *nonnegligible correctness and security error, and even if the obfuscation procedure is inefficient).*

Our result, proven in Section 7.5, uses roughly the same proof strategy as previous works [BGI+01; AF16], but overcomes the two main issues described above as follows:

Homomorphic evaluation. Previous constructions rely on the obfuscator to implement the homomorphic evaluations, by obfuscating the functionality "decrypt, then apply a gate, then re-encrypt". However, by now, we know how to build quantum fully-homomorphic encryption schemes directly [Mah18a; Bra18], based on the learning-with-errors (LWE) assumption. Thus, in our construction, we can remove the homomorphic gate evaluation from the obfuscated circuits: the adversary can do the homomorphic evaluation of the point function herself, using a quantum fully-homomorphic encryption scheme. With the homomorphic evaluation removed from it, the class of circuits that we prove impossible to obfuscate can remain classical.

This solution introduces a slight complication: part of the functionality of the circuit we construct is now to return the public evaluation key. However, unless one is willing to make an assumption on the circular security of the homomorphic encryption, the size of this key (and therefore the size of the circuit) scales with the size of the circuit that needs to be homomorphically evaluated. To get rid of this inconvenient dependence, our unobfuscatable circuit returns the public key in small, individual blocks that can be independently computed. We argue that any classical-key quantum fully-homomorphic encryption scheme has public keys that can be decomposed in this way.

Reusability. The circuits that we consider are classical and deterministic. Therefore, if the interpreter map is run on an obfuscation state ρ for a circuit C, plus a classical input x, then by correctness, the result is (close to) a computational-basis state $|C(x)\rangle$. This output can be copied out to a separate wire without disturbing the state, and the interpreter map can be reversed, recovering the obfuscation ρ to be used again. If the interpreter map is not unitary, then it can be run coherently (i.e., keeping purification registers around instead of measuring wires), and this coherent version can be reversed as long as the purification registers are not measured.

At one point in our proof, we will need to run the interpreter map homomorphically on (an encryption of) ρ and x. This may result in a superposition of different ciphertexts for $C(x)$, which cannot cleanly be copied out to a separate wire without entangling that wire with the output. Thus, recovering ρ is not necessarily possible after the homomorphic-evaluation step.

We circumvent this problem by making sure that the homomorphic evaluation occurs last, so that ρ is not needed anymore afterwards. This reordering is achieved by classically obfuscating the part of the circuit that checks whether a ciphertext decrypts to the value β. That way, this functionality becomes a constant output value that a user can request and store before performing the homomorphic evaluation, and use afterwards. To obfuscate the decryption check, we use a classical vbb obfuscator for compute-and-compare functions, which relies on a variant of the LWE assumption [WZ17; GKW17].

Our impossibility result compares to the classical impossibility result [BGI+01] as follows. First, as mentioned, we extend the realm of impossible obfuscators to include obfuscators that produce a quantum state, rather than a classical circuit. Second, the classical impossibility result is unconditional, whereas we require the (standard) assumption that LWE is hard for quantum adversaries. It may be possible to relax this assumption if ρ can be recovered after the homomorphic evaluation (see Section 7.6.1). Third, the class of classical circuits that cannot be obfuscated is slightly different: in our work, it does not have the homomorphic-evaluation functionality built into it, and is therefore arguably simpler, strengthening the impossibility result. However, we stress that in both works, the unobfuscatable circuit class itself is somewhat contrived: the main implication is that its superclass **P** is unobfuscatable.

As an intermediate result, we show in Section 7.4 that it is impossible to vbb obfuscate even just the class of classical multi-bit-output point functions into a quantum state, if the adversary and simulator have access to auxiliary classical information that contains an encryption of the nonzero input value α and a vbb obfuscation of a function depending on the secret key for that encryption.

Theorem (informal). *If LWE is hard for quantum algorithms, then it is impossible to quantum vbb obfuscate multi-bit-output point functions and the all-zero function under the presence of classical dependent auxiliary information (even with nonnegligible soundness and security error).*

At first glance, that may seem to contradict the constructions of vbb obfuscation of point functions [WZ17; GKW17], which are secure even in the presence of dependent auxiliary information. The crucial difference is that those constructions only allow a limited dependency of the auxiliary information, whereas in our impossibility proof, the dependence is slightly stronger. This subtle difference seems to indicate that the gap between possibility and impossibility of vbb obfuscation is closing.

Independently of this work, Ananth and La Placa [AL20] have concurrently shown the general impossibility of quantum copy-protection, thereby also ruling out quantum obfuscation of classical circuits. Their techniques are very similar to ours, but their adversary is somewhat more powerful in the sense that it is able to completely de-obfuscate the program given non-black-box access. They also present some positive results in their work. Their result relies on the same LWE assumption as ours, but in addition they require that the underlying homomorphic-encryption scheme is circularly secure. We avoid circularity by introducing a notion of decomposable public keys for homomorphic encryption. This technique could potentially be used to remove the circularity assumption from the copy-protection impossibility result [AL20] as well.

7.2 Preliminaries

In this section, we describe several preliminaries that are specific to this chapter. We give the formal definition of (virtual-black-box) obfuscation of circuits, specify the variant of quantum homomorphic encryption we will employ, define compute-and-compare functions, and discuss how to recover the input of a quantum computation in case the output is (close to) classical.

7.2.1 Classical and quantum virtual-black-box obfuscation

We consider so-called *circuit* obfuscators: the functionalities to be hidden are represented by circuits. A virtual-black-box circuit obfuscator hides the functionality in such a way that the obfuscation looks like a "black box": the only way to get information about its functionality is to evaluate it on an input and observe the output.

Definition 7.2.1 ([BGI+01, Definition 2.2])**.** A classical virtual black-box obfuscator for the circuit class $\mathcal{F}$ is a probabilistic algorithm $\mathcal{O}$ such that

1. (polynomial slowdown) For every circuit $C \in \mathcal{F}$, $|\mathcal{O}(C)| = \text{poly}(|C|)$;
2. (functional equivalence) For every circuit $C \in \mathcal{F}$, the string $\mathcal{O}(C)$ describes a circuit that computes the same function as C;
3. (virtual black-box) For any PPT adversary A, there exists a PPT simulator $\mathcal{S}$ such that for all circuits $C \in \mathcal{F}$,

$$\left|\Pr\left[A(\mathcal{O}(C)) = 1\right] - \Pr\left[\mathcal{S}^{C}(1^{|C|}) = 1\right]\right| \leq \text{negl}(|C|).$$

As a variation on the third requirement, one may assume that some auxiliary information (which may depend on the circuit C) is present alongside the obfuscation $\mathcal{O}(C)$. In that case, a simulator with access to that auxiliary information should still be able to simulate the adversary's output distribution:

Definition 7.2.2 ([GK05, Definition 3])**.** A classical virtual black-box obfuscator w.r.t. dependent auxiliary input for a circuit class $\mathcal{F}$ is a probabilistic algorithm $\mathcal{O}$ that satisfies Definition 7.2.1, with the "virtual black-box" property redefined as follows:

3. (virtual black-box) For any PPT adversary A, there exists a PPT simulator $\mathcal{S}$ such that for all circuits $C \in \mathcal{F}$ and all strings $\mathsf{aux} \in \{0,1\}^{\text{poly}(|C|)}$ (which may depend on C),

$$\left|\Pr\left[A(\mathcal{O}(C), \mathsf{aux}) = 1\right] - \Pr\left[A(\mathcal{S}^{C}(1^{|C|}, \mathsf{aux})) = 1\right]\right| \leq \text{negl}(|C|).$$

In the quantum setting, we consider quantum obfuscators for classical circuit classes: that is, the obfuscation $\mathcal{O}(C)$ may be a quantum state. We adapt the definition of quantum obfuscators for quantum circuits by Alagic and Fefferman [AF16, Definition 5].

Definition 7.2.3. A quantum virtual black-box obfuscator for the classical circuit class $\mathcal{F}$ is a quantum algorithm $\mathcal{O}$ and a QPT $\mathcal{J}$ (the "interpreter") such that

1. (polynomial expansion) For every circuit $C \in \mathcal{F}$, $\mathcal{O}(C)$ is an m-qubit quantum state with $m = \text{poly}(n)$;
2. (functional equivalence) For every circuit $C \in \mathcal{F}$ and every input x,
$$\| \mathcal{J}(\mathcal{O}(C) \otimes |x\rangle\langle x|) - |C(x)\rangle\langle C(x)| \|_{\text{tr}} \leqslant \text{negl}(|C|);$$
3. (virtual black-box) For every QPT adversary $\mathcal{A}$, there exists a QPT simulator $\mathcal{S}$ (with superposition access to its oracle) such that for all circuits $C \in \mathcal{F}$,
$$\left|\Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - \Pr[\mathcal{S}^C(1^{|C|}) = 1]\right| \leqslant \text{negl}(|C|).$$

There are a few differences with the classical definition. First, the obfuscation is a quantum state, and not a (classical or quantum) circuit. Second, due to the probabilistic nature of quantum computation, we allow a negligible error in the functional equivalence. Third, the simulator is slightly more powerful because of its superposition access to the functionality of C: a query performs the unitary operation specified by $|x\rangle|z\rangle \mapsto |x\rangle|z \oplus C(x)\rangle$. Note that a quantum adversary can always use a (classical or quantum) obfuscation to compute the obfuscated functionality on a superposition of inputs, obtaining a superposition of outputs. For this reason, the simulator gets superposition access to its oracle in the quantum setting. Throughout this chapter, all oracles supplied to quantum algorithms allow for superposition access.

We can again strengthen the virtual black-box property to include (classical or quantum) dependent auxiliary information: this auxiliary string or state would be provided to both the adversary and the simulator, in the same way as in Definition 7.2.2.

7.2.2 Classical-client quantum fully homomorphic encryption

From the learning-with-errors assumption, it is possible to construct secure QFHE schemes where all client-side operations (key generation, encryption, and decryption) are classical [Mah18a; Bra18]. We slightly adapt Definition 5.2.7 to reflect the fact that ciphertexts are classical. In this chapter, we do not make a distinction between the public key and evaluation key: any information that is necessary for the evaluation is contained in the public key.

Definition 7.2.4. A quantum fully homomorphic encryption scheme QFHE consists of four algorithms, as follows:

- **Key Generation:** $(pk, sk) \leftarrow \mathsf{QFHE.KeyGen}(1^\lambda)$ produces a public key pk and a secret key sk, given a security parameter λ. This is a classical PPT algorithm.

- **Encryption:** $c \leftarrow \mathsf{QFHE.Enc}_{pk}(m)$ encrypts a single-bit message $m \in \{0,1\}$. For multi-bit messages $m \in \{0,1\}^\ell$, we write $\mathsf{QFHE.Enc}_{pk}(m)$ for the bit-by-bit encryption. This algorithm is in general QPT, but it only uses a classical random tape, and furthermore whenever m is classical, so is the encryption algorithm.

- **Decryption:** $m' = \mathsf{QFHE.Dec}_{sk}(c)$ decrypts a ciphertext c into a single-bit message m', using the secret key sk. If c is a ciphertext for a multi-bit message, we write $\mathsf{QFHE.Dec}_{sk}(c)$ for the bit-by-bit decryption. Again this is QPT in general, but it is classical if c is classical.

- **Homomorphic evaluation:** $c' \leftarrow \mathsf{QFHE.Eval}_{pk}(C, c)$ takes as input the public key, a classical description of a BQP circuit C with ℓ input wires and ℓ' output wires, and a bit-by-bit encrypted ciphertext c encrypting ℓ bits. It produces a c', a sequence of ℓ' output ciphertexts. This is a QPT algorithm.

Similarly to the schemes described in Chapters 5 and 6, classical-client QFHE schemes [Mah18a; Bra18] encrypt a message m using a quantum one-time-pad with random keys $a, b \in \{0,1\}$, attaching classical FHE ciphertexts of the one-time pad keys:

$$\mathsf{QFHE.Enc}_{pk}(m) = \mathsf{X}^a\mathsf{Z}^b\,|m\rangle \otimes \left|\mathsf{FHE.Enc}_{pk}(a), \mathsf{FHE.Enc}_{pk}(b)\right\rangle. \tag{7.1}$$

Note that since m is a classical message, this ciphertext can be classically represented as the tuple

$$\left(m \oplus a, \mathsf{FHE.Enc}_{pk}(a), \mathsf{FHE.Enc}_{pk}(b)\right), \tag{7.2}$$

so that encryption may be seen as a classical procedure. Conversely, a classical ciphertext $\tilde{m} \leftarrow \mathsf{FHE.Enc}_{pk}(m)$ can easily be turned into a valid quantum ciphertext by preparing the state $|0\rangle \otimes \left|\tilde{m}, \mathsf{FHE.Enc}_{pk}(0)\right\rangle$, which decrypts to m. Thus, it is possible to freely switch back and forth between quantum and classical ciphertexts, as long as the message is known to be classical.

Extending the encryption procedure from Equation (7.1) to a general quantum state $|\psi\rangle$, rather than a computational-basis state $|m\rangle$, yields the encryption procedure $\mathsf{CL.Enc}$ (see Section 5.3) with quantum ciphertexts. We will use this encryption of quantum states when we supply the encryption of a quantum-state obfuscation as the input to a homomorphic evaluation.

Bootstrapped FHE. Recall Definition 5.2.5, specifying leveled (Q)FHE schemes: the key generation algorithm receives an additional parameter d, and outputs a key set that is suitable for the evaluation of circuits up to depth d. Importantly, the length of sk and the complexity of decryption are independent of d.

We assume, without loss of generality, that the randomness tape used by KeyGen is always of length λ. (If it is not, the length of the randomness tape can be stretched using a pseudorandom generator.)

A common way to construct leveled FHE is via the bootstrapping technique suggested by Gentry [Gen09]. Gentry showed that given a base scheme with homomorphic capacity greater than its decryption depth, it is possible to create a leveled scheme with the following properties.

Definition 7.2.5 (Leveled bootstrapped FHE)**.** A leveled bootstrapped (Q)FHE is constructed using a "base scheme" with key-generation algorithm $\mathsf{SubKeyGen}(1^\lambda)$ (and corresponding encryption, decryption and evaluation algorithms), as follows. The key-generation algorithm $\mathsf{KeyGen}(1^\lambda, 1^d)$ is of the form:

1. Run $\mathsf{SubKeyGen}(1^\lambda)$ with fresh randomness $d+1$ times to generate sub-keys (sk_i, pk_i) for $i = 0, \ldots, d$.
2. Compute $c_i^* \leftarrow \mathsf{Enc}_{pk_i}(sk_{i-1})$ for all $i = 1, \ldots, d$.
3. Output $pk := \big(pk_0, (pk_1, c_1^*), \ldots, (pk_d, c_d^*)\big)$, and $sk := sk_d$.

The decryption algorithm of the bootstrapped scheme is the same as that of the base scheme, and for encryption, only pk_0 is needed.

7.2.3 Point functions and compute-and-compare functions

The class of compute-and-compare functions, as well as its subclass of point functions, plays an important role in this chapter. We define these function classes now.

Definition 7.2.6 (Point function)**.** Let $y \in \{0,1\}^n$. The point function $\mathbf{P}_y$ is defined by

$$\mathbf{P}_y(x) := \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

The value y is called the *target value*. Point functions are a special type of compute-and-compare function, where the function f is the identity:

Definition 7.2.7 (Compute-and-compare function)**.** Let $f : \{0,1\}^m \to \{0,1\}^n$ and $y \in \{0,1\}^n$. The compute-and-compare function $\mathbf{CC}_{f,y}$ is defined by

$$\mathbf{CC}_{f,y}(x) := \begin{cases} 1 & \text{if } f(x) = y \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

One can also consider point functions or compute-and-compare functions with *multi-bit output*: in that case, the function outputs either some string z (instead of 1), or the all-zero string (instead of 0). We denote such functions with $\mathbf{P}_{y,z}$ and $\mathbf{CC}_{f,y,z}$.

7.2.4 Recovering the input of a quantum circuit

We will consider (efficient) quantum operations as (polynomial-size) circuits, consisting of the following set of basic operations: unitary gates from some fixed constant-size gate set, measurements in the computational basis, and initialization of auxiliary wires in the $|0\rangle$ state.

While unitary gates are always reversible by applying their transpose ($U^\dagger U = I$ for any unitary U), measurement gates may not be, as they can possibly collapse a state. However, we can effectively delay all measurements in a circuit C until the very end, as follows. Define U_C as the unitary that computes C *coherently*: that is, for every computational-basis measurement in C on some wire w, U_C performs a CNOT operation from w onto a fresh auxiliary target wire initialized in the state $|0\rangle$. The circuit C is now equivalent to the following operation: initialize all auxiliary target wires in the $|0\rangle$ state[1], apply the unitary U_C, and measure all auxiliary target wires in the computational basis.

In this work, we will encounter circuits C which, for specific inputs, yield a specific state in the computational basis with very high probability. In the proof of the following lemma, we specify how to use coherent computation in order to learn the output value while preserving the input quantum state. It is similar to Aaronson's "Almost As Good As New Lemma" [Aar04].

Lemma 7.2.8. *Let C be a quantum circuit. There exists an input-recovering circuit C_{rec} such that for all inputs ρ_{in}, the following holds: if $\left\| C(\rho_{\mathsf{in}}) - |x\rangle\langle x| \right\|_{\mathrm{tr}} \leqslant \varepsilon$ for some classical string x and some $\varepsilon > 0$, then*

$$\left\| C_{\mathsf{rec}}(\rho_{\mathsf{in}}) - \left(\rho_{\mathsf{in}} \otimes |x\rangle\langle x|\right) \right\|_{\mathrm{tr}} \leqslant 2\varepsilon.$$

Proof. The input-recovering circuit C_{rec} will consist of running C coherently, copying out the output register, and reverting the coherent computation of C. Suppose the circuit C contains k measurement gates, ℓ initializations of wires in the $|0\rangle$ state, and outputs of length n. Define C_{rec} as:

1. Run U_C on input $\rho_{\mathsf{in}} \otimes \left|0^\ell\right\rangle\!\left\langle 0^\ell\right| \otimes \left|0^k\right\rangle\!\left\langle 0^k\right|^M$, where U_C is the unitary that coherently executes C, and M is the register that contains the auxiliary wires for the coherent measurements.

[1]If, apart from the targets of the aforementioned CNOTs, the circuit C contains any other wires that are initialized in the $|0\rangle$ state inside the circuit, those wires are also considered part of the input of the unitary U_C. They should be initialized to $|0\rangle$ here as well.

2. Copy the wires that are supposed to contain the output $C(\rho_{\mathsf{in}})$ into a register Y, initialized to $|0^n\rangle\langle 0^n|$, using CNOTs.
3. Run $U_C^\dagger$ to recover the original input, and discard any auxiliary wires.

To see that C_{rec} acts as promised, let ρ_{in}, x, and ε be such that $\| C(\rho_{\mathsf{in}}) - |x\rangle\langle x| \|_{\mathrm{tr}} \leq \varepsilon$. If ε is small, the CNOT in Step 2 does not create a lot of entanglement, since the control wires are (close to) the computational-basis state $|x\rangle\langle x|$. The output is therefore (almost) perfectly copied out.

More formally, let ρ_{out} denote the state after the application of U_C, that is,

$$\rho_{\mathsf{out}} \otimes \left|0^n\right\rangle\!\left\langle 0^n\right|^Y := U_C \left(\rho_{\mathsf{in}} \otimes \left|0^\ell\right\rangle\!\left\langle 0^\ell\right| \otimes \left|0^k\right\rangle\!\left\langle 0^k\right|^M \otimes\right) U_C^\dagger \otimes \left|0^n\right\rangle\!\left\langle 0^n\right|^Y \tag{7.5}$$

$$= \sum_{m,m' \in \{0,1\}^k} \alpha_m \alpha_{m'}^* \sigma_{m,m'} \otimes |m\rangle \left\langle m'\right|^M \otimes \left|0^n\right\rangle\!\left\langle 0^n\right|^Y, \tag{7.6}$$

where the $\alpha_m \in \mathbb{C}$ are normalization factors, and $\sigma_{m,m'}$ are density matrices.

Note that if we trace out the M and Y registers, the remaining state is $C(\rho_{\mathsf{in}})$. By the assumption that $\| C(\rho_{\mathsf{in}}) - |x\rangle\langle x| \|_{\mathrm{tr}} \leq \varepsilon$ for some string x, it follows that

$$\left\| \rho_{\mathsf{out}} - \sum_{m,m' \in \{0,1\}^k} \alpha_m \alpha_{m'}^* |x\rangle\langle x| \otimes |m\rangle \left\langle m'\right|^M \right\|_{\mathrm{tr}} \leq \varepsilon. \tag{7.7}$$

For the remaining steps, we use the fact that no unitary can increase the trace distance between two quantum states.

Applying the CNOTs in Step 2 results in a state ρ'_{out} such that

$$\left\| \rho'_{\mathsf{out}} - \sum_{m,m' \in \{0,1\}^k} \alpha_m \alpha_{m'}^* |x\rangle\langle x| \otimes |m\rangle \left\langle m'\right|^M \otimes |x\rangle\langle x|^Y \right\|_{\mathrm{tr}} \leq \varepsilon, \tag{7.8}$$

and therefore $\| \rho'_{\mathsf{out}} - \rho_{\mathsf{out}} \otimes |x\rangle\langle x|^Y \|_{\mathrm{tr}} \leq 2\varepsilon$. Finally, applying $U_C^\dagger$ on both sides and discarding the $k+\ell$ auxiliary wires, we get $\| C_{\mathsf{rec}}(\rho_{\mathsf{in}}) - \rho_{\mathsf{in}} \otimes |x\rangle\langle x|^Y \|_{\mathrm{tr}} \leq 2\varepsilon$. □

The specification of C_{rec} is independent of the specific input state ρ_{in}. However, C_{rec} cannot necessarily recover *all* possible inputs ρ_{in}, only those that lead to an almost-classical output.

7.3 (Q)FHE with decomposable public keys

For the purpose of our result in Section 7.5, we will need to obfuscate a class of circuits that allow to (quantumly) homomorphically evaluate operations of arbitrary polynomial depth. We nevertheless wish to rely only on leveled FHE for the sake of minimizing our assumptions. Therefore, we will later define a class of circuits that are a priori polynomially bounded in size, but which are capable of encapsulating public-key generation of a leveled scheme for an arbitrary polynomial depth d. Recall that in a leveled scheme, even just the length of pk may depend on d.

To achieve a circuit size independent of d, we define the notion of (Q)FHE schemes with *decomposable* public key. Intuitively, in such a scheme, the public key can be generated by first generating a sequence of blocks, each of some size independent of d. These blocks can then be combined into the actual pk of the scheme. Crucially, the generation of the blocks can be done in parallel, and the complexity of generating each block (given the security parameter and the random tape) is independent of d. In other words, a decomposable public key can be generated on the fly, involving small "chunks" of computation that are independent of d. Formally, we recall Definition 5.2.5, and define decomposability as follows.

Definition 7.3.1 (Decomposable public key)**.** A leveled (Q)FHE scheme has a *decomposable public key* if there exists a polynomial $K = K(\lambda, d)$ and a polynomial-time deterministic function $\mathsf{BlockGen}(1^\lambda, i, r, r')$ (where $r, r' \in \{0,1\}^\lambda$) that generates classical strings ("blocks") c_i such that the following holds:

1. **Correctness:** there exists a QPT $\mathsf{Assemble}$ such that for all λ, d, r, and r' (letting $K = K(\lambda, d)$) it holds that

$$\mathsf{Assemble}(c_0, c_1, c_2, \ldots, c_K) = pk,$$

where $(pk, sk) = \mathsf{KeyGen}(1^\lambda, 1^d, r)$, and $c_i = \mathsf{BlockGen}(1^\lambda, i, r, r')$ for all i.

2. **Simulatability:** there exists a QPT simulator $\mathcal{S}$ such that for all d and r,

$$\mathcal{S}(1^\lambda, pk) \overset{c}{\approx} (c_0, c_1, c_2, \ldots, c_K),$$

where $(pk, sk) = \mathsf{KeyGen}(1^\lambda, d, r)$, and the distribution on $(c_0, c_1, c_2, \ldots, c_K)$ on the right-hand side is generated by selecting a uniformly random r', and then for all i, setting $c_i = \mathsf{BlockGen}(1^\lambda, i, r, r')$.

We emphasize that in Definition 7.3.1, the randomness strings r and r' are the same for every run of BlockGen. The reason for this choice is twofold. First, with our final goal in mind of obfuscating the BlockGen functionality, we want to avoid having to specify K independent randomness strings (as that would considerably increase the size of the circuit to obfuscate). Second, most FHE schemes require some form of correlation to exist between the different blocks. Thinking of r and r' as short random seeds for a PRF, this correlation can be realized by running the PRF on the same inputs (see, for example, Section 7.3.1).

7.3.1 Instantiation from bootstrapped schemes

Existing QFHE schemes are based on bootstrapping [Mah18a; Bra18] (see Definition 7.2.5). Without affecting security, we can assume that the randomness for the SubKeyGen is sampled using a PRF. That is, the random tape r of KeyGen is used as a seed for a PRF, and for the ith execution of SubKeyGen, we use the randomness $PRF_r(i)$.

For bootstrapped schemes using a PRF, decomposability follows immediately by definition. In this case, we do not even need the extra randomness r' and can simply set $\mathsf{BlockGen}(1^\lambda, i, r, r')$ to be the process that evaluates $PRF_r(i-1)$ and $PRF_r(i)$ to generate random tapes for SubKeyGen, uses this randomness to generate (sk_{i-1}, pk_{i-1}) and (sk_i, pk_i), generates c_i^* based on these values, and outputs (pk_i, c_i^*). In addition, for $i = 0$, it simply computes $PRF_r(0)$, and uses the resulting randomness to generate pk_0.

Lemma 7.3.2. *Bootstrapping-based leveled QFHE schemes with keys generated from a PRF have decomposable public keys.*

Proof. Define $K(\lambda, d) := d$, and $c_0 := pk_0$. For $i > 0$, define the blocks c_i, which are generated by $\mathsf{BlockGen}(1^\lambda, i, r, r')$, as follows:

$$c_i := \left(pk_i, c_i^* = \mathsf{Enc}_{pk_i}(sk_{i-1})\right), \text{ where } (pk_i, sk_i) \leftarrow \mathsf{SubKeyGen}(1^\lambda, PRF_r(i)),$$
$$(pk_{i-1}, sk_{i-1}) \leftarrow \mathsf{SubKeyGen}(1^\lambda, PRF_r(i-1)). \tag{7.9}$$

Note that for public keys of this form, BlockGen does not make use of the additional randomness r'.

The assembly function $\mathsf{Assemble}(c_0, c_1, \ldots, c_d)$ is a straightforward concatenation of all the blocks: $\mathsf{Assemble}(c_0, c_1, \ldots, c_d) := (c_0, c_1, \ldots, c_d)$.

Simulatability as in Definition 7.3.1 is also easily satisfied: a simulator $\mathcal{S}$, for a public key pk and index i, reads out the pair (pk_i, c_i^*). It can thereby exactly produce the list $(c_0, c_1, \ldots, c_d)$. □

7.3.2 Instantiation from any leveled (Q)FHE

We now observe that we can instantiate a (Q)FHE with decomposable public keys from any leveled scheme, even ones that are not bootstrapped.

Decomposing the public key of a general QFHE scheme is done via garbled circuits [Yao86; App17], as we will briefly outline here.

A block c_i corresponds to a single garbled gate of the circuit for KeyGen. That is, $\mathsf{BlockGen}(1^\lambda, i, r, r')$ returns a garbling of the ith gate[2] of $\mathsf{KeyGen}(1^\lambda, d, r)$, using r' as

[2]The total number of blocks, $K(\lambda, d)$, will be the number of gates in $\mathsf{KeyGen}(1^\lambda, 1^d, r)$. Since the number of gates is polynomial in λ, it suffices for the length of the PRF seed r' to be linear in λ.

a PRF seed to generate sufficient randomness for the garbling. A separate block (e.g., c_0) contains the required encoding/decoding information to use the garbled circuit. To assemble the public key, a user concatenates all garbled gates, and evaluates the garbled circuit to obtain the output *pk*. Conversely, by the privacy property of garbled circuits [BHR12], a simulator given only the security parameter λ and the output *pk* of the garbled circuit, can reproduce a garbled circuit that is indistinguishable from the actual garbled circuit. It can then return the gates of that simulated garbled circuit as the blocks c_i.

Any result relying on the decomposability of the public key of a non-bootstrapping based QFHE scheme of course also relies on any computational assumptions required for the security of the garbled-circuit construction.

7.4 Impossibility with respect to dependent auxiliary information

In this section, we show impossibility of virtual-black-box quantum obfuscation of classical point functions under dependent auxiliary information. It sets the stage for our main result, Theorem 7.5.1, where we incorporate the auxiliary information into the circuit, constructing a circuit class which is unobfuscatable even without the presence of any auxiliary information. Although the result in the current section is perhaps less surprising, the proof contains the most important technical details of our work.

The impossibility result requires two cryptographic primitives, both of which can be built from the hardness of LWE: (1) quantum fully homomorphic encryption with classical client-side operations ([Mah18a; Bra18], see Section 7.2.2), and (2) classical vbb obfuscation of compute-and-compare functions [WZ17; GKW17]. Our result therefore holds under the assumption that LWE is hard. The least favorable LWE parameters are required for the obfuscation of compute-and-compare functionalities, and are discussed in Section 7.4.1.

In Section 7.4.1, we describe the classical obfuscator for compute-and-compare functions that we use. We will apply it to a specific class of compute-and-compare functions with a specific type of auxiliary information. In Section 7.4.2, we use this specific application to define a class of circuits and auxiliary-information strings that is unobfuscatable in the quantum vbb sense. The impossibility proof follows in Section 7.4.3.

7.4.1 Classical obfuscation of compute-and-compare functions

Under the assumption that LWE (with polynomial dimension and exponential modulus in the security parameter λ) is hard, it is possible to classically obfuscate compute-

and-compare functions [WZ17; GKW17]. We will write “LWE*” to denote their specific variant of the LWE assumption. We note that LWE is known to be at least as hard as worst-case lattice problems [Reg05; PRS17]. In particular, the aforementioned parameter regime LWE* translates to the worst-case hardness of the Gap Shortest Vector Problem (GapSVP) with sub-exponential approximation factor (in the dimension of the lattice). There is currently no known super-polynomial quantum speedup for GapSVP, and the best known quantum (and classical) algorithms require sub-exponential running time.

Wichs and Zirdelis [WZ17] achieve so-called distributional virtual-black-box obfuscation of compute-and-compare functions $\mathbf{CC}_{f,y}$, assuming that the target value y has sufficient pseudo-entropy given a description of the function f. The obfuscation is even secure in the presence of (dependent) auxiliary information, so long as the pseudo-entropy of the target value remains high, even conditioned on this auxiliary information.

In our construction, we provide a classically-obfuscated compute-and-compare function as auxiliary information to a quantum obfuscation. We will require that the target value of the compute-and-compare function is sufficiently random, even given the rest of the auxiliary information (including the quantum obfuscation).

More specifically, for any leveled quantum homomorphic-encryption scheme (KeyGen, Enc, Eval, Dec), fixed bit string α, depth parameter d, and a classical obfuscation procedure $\mathcal{O}(\cdot)$, define a distribution ensemble $\{D_\lambda^{\alpha,d}\}_{\lambda\in\mathbb{N}}$ that samples

$$\begin{aligned}(pk, \widetilde{\alpha}, o_{sk,\beta}) \leftarrow D_\lambda^{\alpha,d} \quad \text{as} \quad & (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d),\\ & \widetilde{\alpha} \leftarrow \mathsf{Enc}_{pk}(\alpha),\\ & \beta \leftarrow_R \{0,1\}^\lambda,\\ & o_{sk,\beta} \leftarrow \mathcal{O}\left(\mathbf{CC}_{\mathsf{Dec}_{sk},\beta}\right), \end{aligned} \tag{7.10}$$

where $\mathbf{CC}_{\mathsf{Dec}_{sk},\beta}$ is a compute-and-compare function as in Definition 7.2.7. For each α, d, and λ, the target value β is chosen independently of all other information: its pseudo-entropy is λ, even conditioned on $\widetilde{\alpha}$ and Dec_{sk}. Therefore, there exists an obfuscation procedure for this class of compute-and-compare programs that has distributional indistinguishability in the following sense:

Lemma 7.4.1 (Application of [WZ17, Theorem 5.2]). *Under the LWE* assumption, there exists a classical obfuscation procedure* $O_{\mathbf{CC}}(\cdot)$ *and a (nonuniform) simulator* $\mathcal{S}$ *such that for all* α *and* d*,*

$$(pk, \widetilde{\alpha}, o_{sk,\beta}) \overset{c}{\approx} (pk, \widetilde{\alpha}, \mathcal{S}(1^\lambda, \mathsf{params})),$$

where $(pk, \widetilde{\alpha}, o_{sk,\beta}) \leftarrow D_\lambda^{\alpha,d}$ *using* $O_{\mathbf{CC}}(\cdot)$ *as the obfuscation procedure* $\mathcal{O}(\cdot)$*, and* params *is some information that is independent of* sk *and* β *(e.g., it may contain the size of the circuit and/or* λ*).*

In the rest of this chapter, $O_{\mathbf{CC}}(\cdot)$ will implicitly be the obfuscation procedure used in the distributions $D_\lambda^{\alpha,d}$.

We note that strictly speaking, the proof of Lemma 7.4.1 only contains a *classical* reduction from the hardness of distinguishing the aforementioned distributions to the hardness of solving LWE. However, proofs by (either Karp or Turing) classical polynomial-time reduction from A to B imply that any solver for A can be translated into a solver for B with comparable complexity. So in particular, if the solver for A runs in *quantum* polynomial time, then so will the resulting solver for B. Therefore, Lemma 7.4.1 holds for quantum distinguishers as well.

As a consequence of Lemma 7.4.1, we show that it is hard to guess the value of α, given only a ciphertext $\widetilde{\alpha}$ for α, and an obfuscation of the compute-and-compare function. Intuitively, since the information α is completely independent of the target value β, the obfuscation effectively hides the secret key sk that would be necessary to learn α.

Lemma 7.4.2. *Under the LWE* assumption, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that for any QPT algorithm A and any d,*

$$\Pr[A(pk, \widetilde{\alpha}, o_{sk,\beta}) = \alpha] \leqslant \mathrm{negl}(\lambda). \tag{7.11}$$

Here, the probability is over $\alpha \leftarrow_R \{0,1\}^\lambda$, $(pk, \widetilde{\alpha}, o_{sk,\beta}) \leftarrow D_\lambda^{\alpha,d}$, and the execution of A.

Proof. The result follows almost directly from Lemma 7.4.1, except that we want to bound the probability that A outputs the multi-bit string α, whereas Lemma 7.4.1 only deals with algorithms with a single-bit output.

To bridge the gap, define an algorithm A'_α that runs A on its input, and compares the output of A to α: if they are equal, A'_α outputs 1; otherwise, it outputs 0.

For any *fixed* value of α, we have

$$\begin{aligned}\Pr[A(pk, \widetilde{\alpha}, o_{sk,\beta}) = \alpha] &= \Pr[A'_\alpha(pk, \widetilde{\alpha}, o_{sk,\beta}) = 1] && (7.12)\\ &\overset{(*)}{\approx} \Pr[A'_\alpha(pk, \widetilde{\alpha}, \mathcal{S}(1^\lambda, \mathsf{params})) = 1] && (7.13)\\ &= \Pr[A(pk, \widetilde{\alpha}, \mathcal{S}(1^\lambda, \mathsf{params})) = \alpha]. && (7.14)\end{aligned}$$

The approximation (*) follows from Lemma 7.4.1, and holds up to a difference of $\mathrm{negl}(\lambda)$.

To complete the proof, note that $\mathcal{S}(1^\lambda, \mathsf{params})$ depends neither on α nor on sk. Thus, randomizing over α again, and invoking privacy of the encryption, we get

$$\begin{aligned}\Pr[A(pk, \widetilde{\alpha}, o_{sk,\beta}) = \alpha] &\approx \Pr[A(pk, \widetilde{\alpha}, \mathcal{S}(1^\lambda, \mathsf{params})) = \alpha] && (7.15)\\ &\leqslant \mathrm{negl}(|\alpha|) = \mathrm{negl}(\lambda). && (7.16)\end{aligned}$$

□

We have thus established that, even in the presence of an obfuscated compute-and-compare function that depends on the secret key, encryptions remain secure (in the one-way sense). For this security to hold, it is important that the target value β is sufficiently independent of the plaintext α.

7.4.2 An unobfuscatable circuit class

In this subsection, we define the class of circuits and auxiliary-information strings that we will prove unobfuscatable. Like Barak et al. [BGI+01], we will exploit the idea that access to an object (circuit or quantum state) that allows the evaluation of a function is more powerful than mere black-box access to the functionality: in particular, it allows the *homomorphic* evaluation of the function. For this argument to work, it is important that the function is not easily learnable through black-box access. We will use point functions, as in [BGI+01]: with black-box access only, it is hard to tell the difference between a point function and the all-zero function $\mathbf{Z}_\lambda : \{0,1\}^\lambda \to \{0^\lambda\}$, that always returns the all-zero string of length λ.

The circuits in the unobfuscatable class will be accompanied by auxiliary information containing the public key (suitable for homomorphic evaluations up to depth d), an encryption of α, and a classical obfuscation of the compute-and-compare function that decrypts its input and compares the plaintext to β. The depth parameter d is not fixed a priori: instead, different values of d are included into the class of possible auxiliary-information strings. For any obfuscator $(\mathcal{O}_Q(\cdot), \mathcal{J})$, there is a (polynomial-size) auxiliary-information string included in the class that contains a public key suitable for homomorphically evaluating $\mathcal{J}$.

Consider the class $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}^{\mathsf{point}}_{\lambda,d} \cup \mathcal{C}^{\mathsf{zero}}_{\lambda,d}\right)$ of circuits plus auxiliary information, where

$$\mathcal{C}^{\mathsf{point}}_{\lambda,d} := \left\{(\mathbf{P}_{\alpha,\beta}, (pk, \widetilde{\alpha}, o_{sk,\beta})) \;\middle|\; \alpha \in \{0,1\}^\lambda, (pk, \widetilde{\alpha}, o_{sk,\beta}) \in \operatorname{supp}(D^{\alpha,d}_\lambda)\right\}, \tag{7.17}$$

$$\mathcal{C}^{\mathsf{zero}}_{\lambda,d} := \left\{(\mathbf{Z}_\lambda, (pk, \widetilde{\alpha}, o_{sk,\beta})) \;\middle|\; \alpha \in \{0,1\}^\lambda, (pk, \widetilde{\alpha}, o_{sk,\beta}) \in \operatorname{supp}(D^{\alpha,d}_\lambda)\right\}. \tag{7.18}$$

The class $\mathcal{C}^{\mathsf{point}}_{\lambda,d}$ contains all λ-bit point functions, together with an encryption of the point input α, and a function that checks whether a ciphertext decrypts to the target value β. $\mathcal{C}^{\mathsf{zero}}_{\lambda,d}$ contains the all-zero function $\mathbf{Z}_\lambda$ (which is itself a point function), but still with auxiliary information for the possible values of α and β.

Suppose that some quantum obfuscation $(\mathcal{O}_Q(\cdot)), \mathcal{J})$ exists. We define a QPT algorithm A, which expects an obfuscation $\rho = \mathcal{O}_Q(\mathbf{P}_{\alpha,\beta})$ (or $\mathcal{O}_Q(\mathbf{Z}_\lambda)$), together with the classical auxiliary information $\mathsf{aux} = (pk, \widetilde{\alpha}, o_{sk,\beta})$). On general inputs ρ and $\mathsf{aux} = (\mathsf{key}, \mathsf{ctxt}, \mathsf{obf})$ of this form, let A do as follows:

1. Run $\mathsf{QFHE.Eval}_{\mathsf{key}}\left(\mathcal{J}, \mathsf{Enc}_{\mathsf{key}}(\rho) \otimes |\mathsf{ctxt}\rangle\langle\mathsf{ctxt}|\right)$ to homomorphically evaluate the interpreter algorithm $\mathcal{J}$. Let q be the depth of the circuit for $\mathcal{J}$: because

the interpreter is efficient, $q = \mathrm{poly}(\lambda)$. If $d \geqslant q$, $\rho = \mathcal{O}_Q(\mathbf{P}_{\alpha,\beta})$, $\mathsf{key} = pk$, and $\mathsf{ctxt} = \widetilde{\alpha}$, then this step results in an encryption of β with high probability. If $d \geqslant q$, $\rho = \mathcal{O}_Q(\mathbf{Z}_\lambda)$, $\mathsf{key} = pk$, and $\mathsf{ctxt} = \widetilde{\alpha}$, then it results in an encryption of 0^λ. (Note that we use classical and quantum ciphertexts for the QFHE scheme interchangeably here: see Section 7.2.2 for a justification.) If $d < q$, then the public evaluation key is insufficient for the evaluation of $\mathcal{J}$: in that case, output 0 and abort the computation.

2. Run obf on the output of the previous step. If $\mathsf{obf} = o_{sk,\beta}$, this will indicate whether the previous step resulted in a ciphertext for β (in that case, output 1) or not (in that case, output 0).

If $d > q$, then the length of the auxiliary information may be superpolynomial in λ, and the runtime of the above algorithm A may not be polynomial in λ. If $d < q$, then A fails to tell the difference between an element from $\mathcal{C}^{\mathsf{point}}_{\lambda,d}$ and $\mathcal{C}^{\mathsf{zero}}_{\lambda,d}$. However, for our impossibility result, it suffices that A has the following intended behavior for $d = q$. (But note that we cannot define our circuit class to contain only circuits with $d = q$, since q depends on the specific obfuscator.)

A will almost certainly output 1 when given an element from $\mathcal{C}^{\mathsf{point}}_{\lambda,q}$, because of the functional equivalence of the quantum and classical obfuscations, and the correctness of the homomorphic evaluation. Similarly, when given an element from $\mathcal{C}^{\mathsf{zero}}_{\lambda,q} - \mathcal{C}^{\mathsf{point}}_{\lambda,q}$, it will almost certainly output 0. Formally, for all $\alpha, \beta \in \{0,1\}^\lambda - \{0^\lambda\}$, and letting $d = q$,

$$\Pr\left[A(\mathcal{O}_Q(\mathbf{P}_{\alpha,\beta}), pk, \widetilde{\alpha}, o_{sk,\beta}) = 1\right] \geqslant 1 - \mathrm{negl}(\lambda), \tag{7.19}$$

$$\Pr\left[A(\mathcal{O}_Q(\mathbf{Z}_\lambda), pk, \widetilde{\alpha}, o_{sk,\beta}) = 1\right] \leqslant \mathrm{negl}(\lambda). \tag{7.20}$$

The vastly different output distribution of A when given an obfuscation of a point function versus the zero function are due the fact that A has an actual representation of the function to feed into the interpreter $\mathcal{J}$. In the proof in the next subsection, we will see that a simulator, with only black-box access to these functionalities, will not be able to make that distinction.

7.4.3 Impossibility proof

We are now ready to state and prove the impossibility theorem for quantum obfuscation of classical circuits with dependent auxiliary input. We reiterate that the two assumptions (quantum FHE and compute-and-compare obfuscation) can be realized under the LWE* assumption.

Theorem 7.4.3 (Impossibility of quantum obfuscation w.r.t. auxiliary input)**.** *Suppose that a classical-client quantum fully homomorphic encryption scheme* QFHE

exists that satisfies Definition 7.2.4, and a classical obfuscation procedure $O_{\mathbf{CC}}(\cdot)$ *for compute-and-compare functionalities exists that satisfies Lemma 7.4.1. Then any (not necessarily efficient) quantum obfuscator* $(\mathcal{O}_Q(\cdot), \mathcal{J})$ *for the class* $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}^{\mathsf{point}}_{\lambda,d} \cup \mathcal{C}^{\mathsf{zero}}_{\lambda,d}\right)$ *satisfying conditions 1 (polynomial expansion) and 2 (functional equivalence) from Definition 7.2.3 cannot be virtual black-box under the presence of classical dependent auxiliary input, i.e., cannot satisfy condition 3 from Definition 7.2.3 where both* $\mathcal{A}$ *and* $\mathcal{S}$ *get access to a classical string* aux *(which may depend on* C*).*

It may seem that the class $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}^{\mathsf{point}}_{\lambda,d} \cup \mathcal{C}^{\mathsf{zero}}_{\lambda,d}\right)$, consisting of point functions, is classically obfuscatable using $O_{\mathbf{CC}}(\cdot)$ from [WZ17; GKW17]. That obfuscation is secure if α (which is the target value if we view $\mathbf{P}_{\alpha,\beta}$ as the multi-bit output compute-and-compare function $\mathbf{CC}_{\mathsf{id},\alpha,\beta}$) is unpredictable given the auxiliary information $\mathsf{aux} = (pk, \widetilde{\alpha}, o_{sk,\beta})$. On the surface, that seems to be the case: only an encryption of α is available in the auxiliary information. However, the secret key sk is present as part of the compute-and-compare function $\mathbf{CC}_{\mathsf{Dec}_{sk},\beta}$. That function is obfuscated, but the obfuscation is not secure in the presence of (an obfuscation of) $\mathbf{P}_{\alpha,\beta}$. Thus, the compute-and-compare obfuscation results [WZ17; GKW17] *almost* apply to the class $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}^{\mathsf{point}}_{\lambda,d} \cup \mathcal{C}^{\mathsf{zero}}_{\lambda,d}\right)$, but not quite. Hence we are able to prove impossibility of obfuscating it, which we do below.

Proof. The proof structure is similar to the classical impossibility proof [BGI+01], and is by contradiction: assume that a quantum obfuscation $(\mathcal{O}_Q(\cdot), \mathcal{J})$ for the class $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}^{\mathsf{point}}_{\lambda,d} \cup \mathcal{C}^{\mathsf{zero}}_{\lambda,d}\right)$ does exist that satisfies all three conditions. We will show that the output distribution of the algorithm A defined in Section 7.4.2 is approximately the same for every element of the class, contradicting Equations (7.19) and (7.20).

By the assumption of the existence of a secure quantum obfuscation $(\mathcal{O}_Q(\cdot), \mathcal{J})$, there exists a simulator $\mathcal{S}$ such that

$$\left|\Pr[A(\mathcal{O}_Q(\mathbf{P}_{\alpha,\beta}), \mathsf{aux}) = 1] - \Pr[\mathcal{S}^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \mathsf{aux}) = 1]\right| \leq \mathrm{negl}(\lambda), \text{ and} \tag{7.21}$$

$$\left|\Pr[A(\mathcal{O}_Q(\mathbf{Z}_\lambda), \mathsf{aux}) = 1] - \Pr[\mathcal{S}^{\mathbf{Z}_\lambda}(1^\lambda, \mathsf{aux}) = 1]\right| \leq \mathrm{negl}(\lambda). \tag{7.22}$$

Here, the probability is taken over $\alpha \leftarrow_R \{0,1\}^\lambda$ and $\mathsf{aux} = (pk, \widetilde{\alpha}, O_{\mathbf{CC}}(\mathbf{CC}_{\mathsf{Dec}_{sk},\beta})) \leftarrow D^{\alpha,q}_\lambda$, where $q = \mathrm{poly}(\lambda)$ is the depth of the circuit for $\mathcal{J}$. Note that $\mathcal{S}$ does not depend on α, β, sk, or pk.

In the remainder of this proof we show that for any $\mathcal{S}$ (independent of α, β, sk, pk),

$$\left|\Pr[\mathcal{S}^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \mathsf{aux}) = 1] - \Pr[\mathcal{S}^{\mathbf{Z}_\lambda}(1^\lambda, \mathsf{aux}) = 1]\right| \leq \mathrm{negl}(\lambda), \tag{7.23}$$

from which it can be concluded that

$$\left|\Pr[A(\mathcal{O}_Q(\mathbf{P}_{\alpha,\beta}),\mathsf{aux})=1]-\Pr[A(\mathcal{O}_Q(\mathbf{Z}_\lambda),\mathsf{aux})=1]\right|\leqslant \operatorname{negl}(\lambda). \tag{7.24}$$

Since Equations (7.19) and (7.20) imply that this difference must be at least $1-\operatorname{negl}(\lambda)$, Equation (7.24) yields a contradiction.

To show that Equation (7.23) holds, i.e., to bound the difference in output probabilities of $\mathcal{S}$ when given an oracle for $\mathbf{P}_{\alpha,\beta}$ versus an oracle for $\mathbf{Z}_\lambda$, we employ the one-way to hiding lemma (Lemma 2.4.4). It says that there exists a QPT algorithm B such that

$$\left|\Pr[\mathcal{S}^{\mathbf{P}_{\alpha,\beta}}(1^\lambda,\mathsf{aux})=1]-\Pr[\mathcal{S}^{\mathbf{Z}_\lambda}(1^\lambda,\mathsf{aux})=1]\right|\leqslant 2d'\cdot\sqrt{\Pr[B^{\mathbf{Z}_\lambda}(1^\lambda,\mathsf{aux})=\alpha]}, \tag{7.25}$$

where $d'=\operatorname{poly}(\lambda)$ is the query depth of $\mathcal{S}$. However, by Lemma 7.4.2, the probability that B outputs α when given the auxiliary information $\mathsf{aux}=(pk,\widetilde{\alpha},o_{sk,\beta})$ is negligible in λ. Granting B access to the zero-oracle and the additional input 1^λ does not increase this probability, since the value of λ can already be deduced from aux.

We can thus conclude that the difference in Equation (7.25) is negligible, and Equation (7.23) holds, as desired. □

We end this section with a few remarks: we describe some variants and generalizations of Theorem 7.4.3 which almost immediately follow from the presented proof.

Remark. The proof for Theorem 7.4.3 also works if we replace $O_{\mathbf{CC}}(\mathbf{CC}_{\mathsf{Dec}_{sk},\beta})$ inside the distributions $D_\lambda^{\alpha,d}$ with $\mathcal{O}_Q(\mathbf{CC}_{\mathsf{Dec}_{sk},\beta})$, the quantum obfuscation we get from the assumption. This adaptation renders a quantum obfuscator for point functions impossible with respect to dependent auxiliary *quantum* input: a slightly weaker statement, but it does not require the existence of a classical obfuscator for compute-and-compare programs. In particular, the required LWE parameters are better, because we only need the assumption of quantum fully homomorphic encryption.

Remark. Even a quantum obfuscator $(\mathcal{O}_Q(\cdot),\mathcal{J})$ for $\bigcup_{d\in[2^\lambda]}\left(\mathcal{C}_{\lambda,d}^{\mathsf{point}}\cup\mathcal{C}_{\lambda,d}^{\mathsf{zero}}\right)$ with non-negligible errors in the functional equivalence and/or the virtual-black-box property would lead to a contradiction in the proof of Theorem 7.4.3. Concretely, let ε_f denote the error for functional equivalence, and ε_s denote the error for security in the virtual-black-box sense (they are both $\operatorname{negl}(|C|)=\operatorname{negl}(\lambda)$ in Definition 7.2.3). The impossibility proof works for any values of $\varepsilon_f,\varepsilon_s$ such that $\varepsilon_f+\varepsilon_s\leqslant 1/2-1/\operatorname{poly}(\lambda)$. So in particular, even a quantum obfuscator with small constant (instead of negligible) errors in both conditions cannot exist.

7.5 Impossibility without auxiliary information

In this section, we will show that quantum virtual-black-box obfuscation of classical circuits is impossible even when no auxiliary information is present. We will rely heavily on the class constructed in Section 7.4, essentially showing how the auxiliary information can be absorbed into the obfuscated circuit. As a result, the unobfuscatable circuit class itself becomes perhaps less natural, but still consists of classical polynomial-size circuits. Thus, our theorem implies impossibility of quantum vbb obfuscation of the class of all efficient classical circuits.

We would like to consider circuits of the following form:

$$C_{\alpha,\beta,\mathsf{aux}}(b,x) := \begin{cases} \mathsf{aux} = (pk, \widetilde{\alpha}, o_{sk,\beta}) & \text{if } b = 0 \\ \mathbf{P}_{\alpha,\beta}(x) & \text{if } b = 1, \end{cases} \tag{7.26}$$

where $(pk, \widetilde{\alpha}, o_{sk,\beta})$ is generated from $D_{\lambda}^{\alpha,d}$, as in Section 7.4. The input bit b is a choice bit: if it is set to 1, the function $\mathbf{P}_{\alpha,\beta}$ (or $\mathbf{Z}_{\lambda}$) is evaluated on the actual input x, whereas if it is set to 0, the auxiliary information is retrieved.

The idea would then be to retrieve the auxiliary information, followed by a homomorphic evaluation of the branch for $b = 1$. There is a problem with this approach, however: since the auxiliary information aux contains the public evaluation key pk, the circuit C grows with the length of pk. But as the circuit grows, a (non-circularly-secure) QFHE scheme may require a larger depth parameter d, and thereby a longer pk, to perform all evaluation steps, which then causes the circuit to grow, et cetera.

To get around this issue, the unobfuscatable circuit will generate the public key step-by-step, in a construction inspired by Canetti et al. [CLTV15]. We will assume that the public key of the leveled QFHE scheme is decomposable in the sense of Definition 7.3.1.

Given a scheme with a decomposable public key, we redefine the unobfuscatable circuit class as follows. Instead of returning the entire public key at once, the circuit allows the user to request individual blocks c_i, up to some depth d. An honest user can run the circuit $K+1 = K(d,\lambda)+1$ times to obtain $pk = \mathsf{Assemble}(c_0, c_1, \ldots, c_K)$. Again, the depth d will not be fixed a priori, although it will be (exponentially) upper bounded: the circuit will only be able to handle inputs i where $|i| \leq \lambda$. Thus, only up to 2^{λ} blocks c_i can be retrieved.

$$\hat{C}_{\alpha,\beta,d,r,r',\widetilde{\alpha},o_{sk,\beta}}(b,x) := \begin{cases} (\widetilde{\alpha}, o_{sk,\beta}) & \text{if } b = 0, \\ \mathsf{BlockGen}(1^{\lambda}, x, r, r') & \text{if } b = 1 \text{ and } x \leq K(d,\lambda), \\ \perp & \text{if } b = 1 \text{ and } x > K(d,\lambda), \\ \mathbf{P}_{\alpha,\beta}(x) & \text{if } b = 2. \end{cases} \tag{7.27}$$

or

$$\hat{C}'_{\alpha,\beta,d,r,r',\widetilde{\alpha},o_{sk,\beta}}(b,x) := \begin{cases} (\widetilde{\alpha}, o_{sk,\beta}) & \text{if } b = 0, \\ \mathsf{BlockGen}(1^{\lambda}, x, r, r') & \text{if } b = 1 \text{ and } x \leq K(d,\lambda), \\ \perp & \text{if } b = 1 \text{ and } x > K(d,\lambda), \\ \mathbf{Z}_{\lambda}(x) & \text{if } b = 2. \end{cases} \tag{7.28}$$

The first input b is now a choice trit: depending on its value, a different branch of the circuit is executed.

We alter the distribution $D_\lambda^{\alpha,d}$ from Equation (7.10), so that it does not explicitly generate the public key anymore. That information is now generated on-the-fly by setting $b = 1$. The public and secret key are deterministically computed using r to generate the auxiliary information $(\widetilde{\alpha}, o_{sk,\beta})$ for $b = 0$. Consider the distribution ensemble $\{D_\lambda^{\alpha,d,r}\}_{\lambda\in\mathbb{N}}$, where

$$(\widetilde{\alpha}, o_{sk,\beta}) \leftarrow D_\lambda^{\alpha,d,r} \quad \text{as} \quad \begin{aligned} &(pk, sk) = \mathsf{KeyGen}(1^\lambda, 1^d, r), \\ &\widetilde{\alpha} \leftarrow \mathsf{Enc}_{pk}(\alpha), \\ &\beta \leftarrow_R \{0,1\}^\lambda, \\ &o_{sk,\beta} \leftarrow \mathbf{CC}_{\mathsf{Dec}_{sk},\beta}. \end{aligned} \tag{7.29}$$

Note that the value of d does not influence the size of $\widetilde{\alpha}$ or $o_{sk,\beta}$ (and thereby the circuit size of $\hat{C}$ or $\hat{C}'$).

We can then define the following parametrized circuit classes:

$$\hat{\mathcal{C}}_{\lambda,d}^{\mathsf{point}} := \left\{ \hat{C}_{\alpha,\beta,d,r,r',\widetilde{\alpha},o_{sk,\beta}} \;\middle|\; \alpha \in \{0,1\}^\lambda, r, r' \in \{0,1\}^\lambda, (\widetilde{\alpha}, o_{sk,\beta}) \in \operatorname{supp}(D_\lambda^{\alpha,d,r}) \right\}, \tag{7.30}$$

$$\hat{\mathcal{C}}_{\lambda,d}^{\mathsf{zero}} := \left\{ \hat{C}'_{\alpha,\beta,d,r,r',\widetilde{\alpha},o_{sk,\beta}} \;\middle|\; \alpha \in \{0,1\}^\lambda, r, r' \in \{0,1\}^\lambda, (\widetilde{\alpha}, o_{sk,\beta}) \in \operatorname{supp}(D_\lambda^{\alpha,d,r}) \right\}. \tag{7.31}$$

Define the circuit class $\hat{\mathcal{C}}_\lambda^{\mathsf{point}} \cup \hat{\mathcal{C}}_\lambda^{\mathsf{zero}}$, where $\hat{\mathcal{C}}_\lambda^{\mathsf{point}} := \bigcup_{d:K(d,\lambda)\leqslant 2^\lambda} \hat{\mathcal{C}}_{\lambda,d}^{\mathsf{point}}$ and similarly $\hat{\mathcal{C}}_\lambda^{\mathsf{zero}} := \bigcup_{d:K(d,\lambda)\leqslant 2^\lambda} \hat{\mathcal{C}}_{\lambda,d}^{\mathsf{zero}}$. Note that in all circuits in this class, the "auxiliary information" $(\widetilde{\alpha}, o_{sk,\beta})$ is fixed. Hence, when the obfuscation of the compute-and-compare function is requested by setting $b = 0$, the circuit always returns the same obfuscation that depends on the same secret key sk.

Similarly to the setting with auxiliary input, there exists a QPT algorithm A' that has significantly different output distributions when given a circuit from $\hat{\mathcal{C}}_{\lambda,d}^{\mathsf{point}}$ versus a circuit from $\hat{\mathcal{C}}_{\lambda,d}^{\mathsf{zero}}$, for the right value of d. On an input state ρ, we define A' as follows:

1. Run $\mathcal{J}_{\mathsf{rec}}\left(\rho, |b=0\rangle\langle b=0| \otimes |0^\lambda\rangle\langle 0^\lambda|\right)$, where $\mathcal{J}_{\mathsf{rec}}$ is the input-recovering version of the interpreter circuit (see Lemma 7.2.8). If ρ is an obfuscation of a circuit in $\hat{\mathcal{C}}^{\mathsf{point}} \cup \hat{\mathcal{C}}^{\mathsf{zero}}$, this will result in a state (negligibly close to) $\rho \otimes |\widetilde{\alpha}\rangle\langle\widetilde{\alpha}| \otimes |o_{sk,\beta}\rangle\langle o_{sk,\beta}|$. Measure the second and third registers to obtain $\widetilde{\alpha}$ and $o_{sk,\beta}$.

2. Let q be the depth of the interpreter circuit $\mathcal{J}$. Because the interpreter should be efficient, $q = \operatorname{poly}(\lambda)$. Sequentially run $\mathcal{J}_{\mathsf{rec}}(\rho, |b=1\rangle\langle b=1| \otimes |i\rangle\langle i|)$ for all $0 \leqslant i \leqslant K = K(q,\lambda)$ to obtain $(c_0, c_1, \ldots, c_K)$, and compute the public evaluation key $pk = \mathsf{Assemble}(c_0, c_1, \ldots, c_K)$, suitable for homomorphic evaluations of up to depth q. Note that the key pk is only revealed in its entirety if the given circuit

has parameter $d = q$. If $d < q$, A' will notice that $\perp$ is returned for some queries, and outputs 0 at this point.

3. Run $\mathsf{QFHE.Eval}_{pk}\big(\mathcal{J}, \mathsf{Enc}_{pk}(\rho) \otimes \big|\mathsf{Enc}_{pk}(b=2)\big\rangle\big\langle \mathsf{Enc}_{pk}(b=2)\big| \otimes |\tilde{\alpha}\rangle\langle\tilde{\alpha}|\big)$. As in Section 7.4.2, this will result in a ciphertext for β (if ρ was an obfuscation of a circuit in $\hat{\mathcal{C}}^{\mathsf{point}}_{\lambda}$) or a ciphertext for 0^{λ} (if ρ was an obfuscation of a circuit in $\hat{\mathcal{C}}^{\mathsf{zero}}_{\lambda}$), provided that $d = q$.
4. Run $o_{sk,\beta}$ on the output of the previous step. Doing so will indicate whether the previous step resulted in a ciphertext for β or not. If yes, output 1; otherwise, output 0.

Let $(O_Q(\cdot), \mathcal{J})$ be an obfuscator. The algorithm A', when given a random obfuscated circuit from $\hat{\mathcal{C}}^{\mathsf{point}}_{\lambda,q}$, will almost certainly output 1, where q is the depth of $\mathcal{J}$. At the same time, an element from $\hat{\mathcal{C}}^{\mathsf{zero}}_{\lambda,q} - \hat{\mathcal{C}}^{\mathsf{point}}_{\lambda,q}$ will almost certainly result in the output 0. More formally, for all $\alpha, r \in \{0,1\}^{\lambda}$ and $d = q$,

$$\Pr\left[A'(\mathcal{O}_Q(\hat{C}_{\alpha,\beta,d,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1\right] \geqslant 1 - \mathrm{negl}(\lambda), \tag{7.32}$$

$$\Pr\left[A'(\mathcal{O}_Q(\hat{C}'_{\alpha,\beta,d,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1\right] \leqslant \mathrm{negl}(\lambda). \tag{7.33}$$

The probability is taken over $D^{\alpha,d,r}_{\lambda}$, r', and the internal randomness of A'. Compare these inequalities to Equations (7.19) and (7.20).

We are now ready to state our main theorem.

Theorem 7.5.1 (Impossibility of quantum obfuscation)**.** *Suppose that a classical-client quantum fully homomorphic encryption scheme* QFHE *exists that satisfies Definitions 7.2.4 and 7.3.1, and a classical obfuscation procedure* $O_{\mathsf{CC}}(\cdot)$ *for compute-and-compare functionalities exists that satisfies Lemma 7.4.1. Then any (not necessarily efficient) quantum obfuscator* $(\mathcal{O}_Q(\cdot), \mathcal{J})$ *for the class* $\hat{\mathcal{C}}^{\mathsf{point}}_{\lambda} \cup \hat{\mathcal{C}}^{\mathsf{zero}}_{\lambda}$ *satisfying conditions 1 (polynomial expansion) and 2 (functional equivalence) from Definition 7.2.3 cannot be virtual black-box, i.e., cannot satisfy condition 3 from Definition 7.2.3.*

Corollary 7.5.2. *If the LWE* assumption holds, the class of classical polynomial-size circuits cannot be quantum virtual-black-box obfuscated in the sense of Definition 7.2.3.*

Proof of Theorem 7.5.1. We again prove the statement by contradiction, assuming that there does exist an obfuscator $(\mathcal{O}_Q(\cdot), \mathcal{J})$ that securely obfuscates $\hat{\mathcal{C}}^{\mathsf{point}}_{\lambda} \cup \hat{\mathcal{C}}^{\mathsf{zero}}_{\lambda}$. Let q be the depth of $\mathcal{J}$, so that $K(q,\lambda)$ is the number of blocks c_i of the evaluation key required by A' to successfully distinguish between an element of $\hat{\mathcal{C}}^{\mathsf{point}}_{\lambda,q}$ and of $\hat{\mathcal{C}}^{\mathsf{zero}}_{\lambda,q}$.

By the assumption that $(\mathcal{O}_Q(\cdot),\mathcal{J})$ is secure, there must exist a simulator $\mathcal{S}_0$ such that for all $\alpha, r \in \{0,1\}^\lambda$ (and setting $d = q$),

$$\left|\Pr[A'(\mathcal{O}_Q(\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1] - \Pr[\mathcal{S}_0^{\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}}(1^\lambda) = 1]\right| \leqslant \mathrm{negl}(\lambda), \tag{7.34}$$

$$\left|\Pr[A'(\mathcal{O}_Q(\hat{C}'_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1] - \Pr[\mathcal{S}_0^{\hat{C}'_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}}(1^\lambda) = 1]\right| \leqslant \mathrm{negl}(\lambda). \tag{7.35}$$

The probabilities are taken over $(\tilde{\alpha}, o_{sk,\beta}) \leftarrow D_\lambda^{\alpha,d,r}$ and $r' \leftarrow_R \{0,1\}^\lambda$, and the internal randomness of A' and $\mathcal{S}_0$.

The output distribution of $\mathcal{S}_0$ can be exactly simulated by another simulator, $\mathcal{S}_1$, that only has access to an oracle for $\mathbf{P}_{\alpha,\beta}$ or $\mathbf{Z}_\lambda$, and gets the auxiliary information pk, $\tilde{\alpha}$, and $o_{sk,\beta}$ as input. $\mathcal{S}_1$ can simply run $\mathcal{S}_0$, simulating each oracle query using its own oracle, auxiliary input, or a combination thereof. If (part of) the query of $\mathcal{S}_0$ is for some block c_i, $\mathcal{S}_1$ can use the decomposability of pk to compute the individual blocks. We formally show the existence of such an $\mathcal{S}_1$ in Corollary 7.5.4 below.

We can thus conclude that for all $\alpha, r \in \{0,1\}^\lambda$,

$$\left|\Pr[A'(\mathcal{O}_Q(\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1] - \Pr[\mathcal{S}_1^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, pk) = 1]\right| \leqslant \mathrm{negl}(\lambda), \tag{7.36}$$

$$\left|\Pr[A'(\mathcal{O}_Q(\hat{C}'_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1] - \Pr[\mathcal{S}_1^{\mathbf{Z}_\lambda}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, pk) = 1]\right| \leqslant \mathrm{negl}(\lambda). \tag{7.37}$$

Again, the probabilities are over $D_\lambda^{\alpha,d,r}$ and r', A', and $\mathcal{S}_1$.

However, by Equation (7.23) in the proof of Theorem 7.4.3, the output distribution of $\mathcal{S}_1$ can only differ negligibly between the two different oracles. Thus, we have

$$\left|\Pr[A'(\mathcal{O}_Q(\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1] - \Pr[A'(\mathcal{O}_Q(\hat{C}'_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}})) = 1]\right| \leqslant \mathrm{negl}(\lambda). \tag{7.38}$$

This contradicts our observation that on input $D_{\alpha,\beta,k,q,\mathsf{aux}}$, A' will almost always output 1, whereas on input $D_{\alpha,\beta,k,q,\mathsf{aux}}$, it will almost always output 0.

This contradicts our observation in Equations (7.32) and (7.33) that on input $\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}$, A' will almost always output 1, whereas on input $\hat{C}'_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}$, it will almost always output 0. □

We end this chapter with an auxiliary lemma, and its corollary that was used in the proof of Theorem 7.5.1.

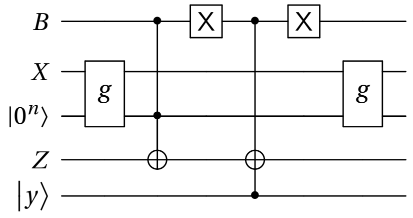
Lemma 7.5.3. *Let $g : \{0,1\}^m \to \{0,1\}^n$ for $m,n \in \mathbb{N}$, and let $y \in \{0,1\}^n$. Let $f : \{0,1\} \times \{0,1\}^m \to \{0,1\}^n$ be defined by*

$$f(b,x) := \begin{cases} y & \text{if } b = 0 \\ g(x) & \text{if } b = 1. \end{cases} \tag{7.39}$$

Then for every QPT $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all f, g of the form described above, and all input states ρ:

$$\Pr[\mathcal{A}^f(\rho) = 1] \;=\; \Pr[\mathcal{S}^g(\rho, y) = 1]. \tag{7.40}$$

Proof. Recall that since $\mathcal{A}$ and $\mathcal{S}$ are quantum algorithms, they access their oracles in superposition: that is, $\mathcal{A}$ has access to the map defined by $|x\rangle|z\rangle \mapsto |x\rangle\left|z \oplus f(x)\right\rangle$, and S has access to the map defined by $|x\rangle|z\rangle \mapsto |x\rangle\left|z \oplus g(x)\right\rangle$. The simulator $\mathcal{S}$ runs $\mathcal{A}$ on input ρ, and simulates any oracle calls to f (on inputs registers BX and output register Z) using two oracle calls to g. It only needs to prepare an auxiliary register in the state $|0^n\rangle$, and run the following circuit:



(7.41)

To see that this circuit exactly simulates a query to f on BXZ, consider an arbitrary query state

$$\sum_i \alpha_i |b_i, x_i\rangle^{BX} |z_i\rangle^{Z} \left|\varphi_i\right\rangle^{R}, \tag{7.42}$$

where R is some purifying register. The state on $BXZR$ (plus the two auxiliary registers containing $|0^n\rangle$ and $\left|y\right\rangle$) after the above circuit is executed, is equal to

$$\sum_i \alpha_i |b_i, x_i\rangle^{XB} \left|0^n\right\rangle \left|z_i \oplus b \cdot g(x_i) \oplus (1-b) \cdot y\right\rangle^{Z} \left|y\right\rangle \left|\varphi_i\right\rangle^{R} \tag{7.43}$$

$$= \sum_i \alpha_i |b_i, x_i\rangle^{XB} \left|0^n\right\rangle \left|z_i \oplus f(x_i)\right\rangle^{Z} \left|y\right\rangle \left|\varphi_i\right\rangle^{R}, \tag{7.44}$$

which is exactly the state that would result from a direct query to f. □

Corollary 7.5.4. *Let $\hat{\mathcal{C}}_\lambda^{\mathsf{point}}$ be as in Equation* (7.30)*, and q be as in the definition of A' below it. Then for any QPT $\mathcal{S}_0$, there exists a QPT simulator $\mathcal{S}_1$ such that for all $\alpha, r \in \{0,1\}^\lambda$,*

$$\left|\Pr[\mathcal{S}_0^{\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}}(1^\lambda) = 1] - \Pr[\mathcal{S}_1^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, pk) = 1]\right| \leq \mathrm{negl}(\lambda). \tag{7.45}$$

A similar statement holds for circuits from $\hat{\mathcal{C}}_\lambda^{\mathsf{zero}}$.

Proof. The statement is proven via an intermediate simulator $\mathcal{S}_2$. This simulator is constructed by repeated application of Lemma 7.5.3, so that for all α, r,

$$\left|\Pr[\mathcal{S}_0^{\hat{C}_{\alpha,\beta,q,r,r',\tilde{\alpha},o_{sk,\beta}}}(1^\lambda) = 1] - \Pr[\mathcal{S}_2^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, c_0, c_1, c_2, \dots, c_K, \perp) = 1]\right| \leqslant \text{negl}(\lambda), \tag{7.46}$$

where $K = K(q, \lambda)$ as in Definition 7.3.1. On the right-hand side, the probability is additionally over a random choice of r' (resulting in the sequence $(c_0, c_1, c_2, \dots, c_K)$), representing the internal randomness of $\mathcal{S}_2$.

Next, we apply the simulatability property of Definition 7.3.1. It states that there exists a simulator $\mathcal{S}_3$ that, given a public key, can generate the distribution over $(c_0, c_1, c_2, \dots, c_K)$ itself. Define

$$\mathcal{S}_1^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, pk) := \mathcal{S}_2^{\mathbf{P}_{\alpha,\beta}}(1^\lambda, \tilde{\alpha}, o_{sk,\beta}, \mathcal{S}_3(pk), \perp), \tag{7.47}$$

and the corollary follows. □

7.6 Conclusion

In this chapter, we answered an open question posed by Alagic and Fefferman [AF16] in the negative: under the assumption that a variant of the learning-with-errors problem is hard for quantum computers, it is not possible to quantumly vbb obfuscate the general class of classical circuits.

We achieved this impossibility result by adapting Barak et al.'s proof for the classical case. We observed that even if the obfuscation is a quantum state, and therefore in principle not reusable, we *can* use it to evaluate the obfuscated multiple (unencrypted) inputs, since the output is almost deterministic. This reusability trick does not work if the output is the result of a homomorphic evaluation: although the plaintext is deterministic, the ciphertext may not be. We used a classical obfuscation of compute-and-compare functions to ensure that the homomorphic evaluation can happen last, so that reusability is not required.

We showed impossibility in two steps: first, we showed that it is impossible to obfuscate compute-and-compare functions *in the presence of dependent auxiliary information.* Then, to move the auxiliary information inside the circuit class, we used the fact that the public key of a classical-client QFHE scheme is decomposable into blocks of size independent of the evaluation depth.

7.6.1 Future directions

The strongest assumption in our work is the existence of the classical vbb obfuscator for compute-and-compare functions, which relies on a variant of LWE. It is necessary because the QFHE evaluation may destroy the obfuscation state when the superposition of output ciphertexts is measured. However, it is not clear if this measurement actually destroys any information on the *plaintext* level, since the plaintext value is deterministic. Thus, it may be possible to recover the (plaintext) obfuscation state after the QFHE evaluation. In that case, it is not necessary to classically obfuscate the compute-and-compare function: it can simply be part of the quantum-obfuscated functionality.

Other open questions are about possibilities rather than impossibilities. What circuit classes *can* be vbb obfuscated into quantum states? Is quantum vbb obfuscation stronger than classical vbb obfuscation, in the sense that it can obfuscate circuit classes that classical vbb cannot? Also, the weaker notion of indistinguishability obfuscation (iO) (also introduced by Barak et al. [BGI+01]) is not affected by our impossibility result: it may still be possible to classically or quantumly iO obfuscate classical functionalities. Could such a construction be lifted into the quantum realm, so that we can (quantum) iO obfuscate quantum functionalities?

Bibliography

[Aar04] Scott Aaronson. “Limitations of quantum advice and one-way communication”. In *Proceedings of the 19th IEEE Annual Conference on Computational Complexity*. 2004, pp. 320–332. DOI: 10.1109/ccc.2004.1313854 (cit. on p. 247).

[ABG+20] Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. “Post-quantum multi-party computation in constant rounds”. 2020. arXiv: 2005.12904 (cit. on p. 91).

[AB08] Dorit Aharonov and Michael Ben-Or. “Fault-tolerant quantum computation with constant error rate”. *SIAM Journal on Computing* (2008), pp. 1207–1282. DOI: 10.1137/s0097539799359385 (cit. on p. 50).

[ABE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. “Interactive proofs for quantum computations”. In *Proceedings of the 1st Symposium on Innovations in Computer Science (ICS)*. 2010, pp. 453–469 (cit. on pp. 7, 265).

[ABEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. “Interactive proofs for quantum computations”. Updated version of [ABE10]. 2017. arXiv: 1704.04487 (cit. on pp. 23, 28, 29, 33, 42–44, 50, 51, 145, 146).

[AAA+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. “Status report on the second round of the NIST post-quantum cryptography standardization process”. *NIST Internal Report 8309*. National Institute of Standards and Technology, July 2020. DOI: 10.6028/NIST.IR.8309 (cit. on p. 5).

[ABDS20] Gorjan Alagic, Zvika Brakerski, Yfke Dulek, and Christian Schaffner. “Impossibility of quantum virtual black-box obfuscation of classical circuits”. 2020. arXiv: 2005.06432 (cit. on p. ix).

[ABF+16] Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. “Computational security for quantum encryption”. In *Proceedings of the 9th International Conference on Information Theoretic Security (ICITS)*. Springer, 2016, pp. 47–71. DOI: 10.1007/978-3-319-49175-2_3 (cit. on pp. 29, 147, 153, 161, 164, 175).

[ACGH19] Gorjan Alagic, Andrew M. Childs, Alex B. Grilo, and Shih-Han Hung. “Non-interactive classical verification of quantum computation”. 2019. arXiv: 1911.08101 (cit. on p. 147).

[ADSS17] Gorjan Alagic, Yfke Dulek, Christian Schaffner, and Florian Speelman. “Quantum fully homomorphic encryption with verification”. In *Advances in Cryptology – ASIACRYPT*. Springer, 2017, pp. 438–467. DOI: 10.1007/978-3-319-70694-8_16 (cit. on pp. ix, 33).

[AF16] Gorjan Alagic and Bill Fefferman. “On quantum obfuscation”. 2016. arXiv: 1602.01771 (cit. on pp. 234, 240, 241, 244, 263).

[AGM18] Gorjan Alagic, Tommaso Gagliardoni, and Christian Majenz. “Unforgeable quantum encryption”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2018, pp. 489–519. DOI: 10.1007/978-3-540-30576-7_21 (cit. on pp. 29, 33, 37, 39, 45, 82).

[AM17] Gorjan Alagic and Christian Majenz. “Quantum non-malleability and authentication”. In *Advances in Cryptology – CRYPTO*. Springer, 2017, pp. 310–341. DOI: 10.1007/978-3-319-63715-0_11 (cit. on pp. 35, 37, 44).

[AM18] Gorjan Alagic and Christian Majenz. Personal communication. Jan. 2018 (cit. on p. 38).

[ABF+13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. “On the relationship between functional encryption, obfuscation, and fully homomorphic encryption”. In *Proceedings of the 14th IMA International Conference on Cryptography and Coding*. Springer, 2013, pp. 65–84. DOI: 10.1007/978-3-642-45239-0_5 (cit. on p. 234).

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. “Quantum security proofs using semi-classical oracles”. In *Advances in Cryptology – CRYPTO*. Springer, 2019, pp. 269–295. DOI: 10.1007/978-3-030-26951-7_10 (cit. on p. 27).

[AMTW00] Andris Ambainis, Michele Mosca, Alain Tapp, and Ronald de Wolf. “Private quantum channels”. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2000, pp. 547–553. DOI: 10.1109/sfcs.2000.892142 (cit. on pp. 5, 29).

[AL20] Prabhanjan Ananth and Rolando La Placa. “Secure software leasing”. 2020. arXiv: 2005.05289 (cit. on pp. 242, 243).

[App17] Benny Applebaum. “Garbled circuits as randomized encodings of functions: a primer”. *Tutorials on the Foundations of Cryptography.* Information Security and Cryptography (ISC) (2017), pp. 1–44. DOI: 10.1007/978-3-319-57048-8_1 (cit. on p. 250).

[AG19] Rotem Arnon-Friedman and Ayal Green. Personal communication. Mar. 2019 (cit. on p. 47).

[AS98] Sanjeev Arora and Shmuel Safra. “Probabilistic checking of proofs: a new characterization of NP”. *Journal of the ACM (JACM)* 45.1 (1998), pp. 70–122. DOI: 10.1145/273865.273901 (cit. on p. 189).

[AS06] Pablo Arrighi and Louis Salvail. “Blind quantum computation”. *International Journal of Quantum Information* 4.05 (2006), pp. 883–898. DOI: 10.1142/s0219749906002171 (cit. on p. 145).

[BB12] Boaz Barak and Zvika Brakerski. “Windows on theory: the swiss army knife of cryptography”. 2012. URL: https://windowsontheory.org/2012/05/01/the-swiss-army-knife-of-cryptography/ (visited on 08/29/2020) (cit. on p. 145).

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. “On the (im)possibility of obfuscating programs”. In *Advances in Cryptology – CRYPTO.* Springer, 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1 (cit. on pp. 239–243, 254, 256, 264, 293, 296).

[BCG+02] Howard Barnum, Claude Crépeau, Daniel Gottesman, Adam Smith, and Alain Tapp. “Authentication of quantum messages”. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 2002, pp. 449–458. DOI: 10.1109/SFCS.2002.1181969 (cit. on pp. 33, 34, 36, 40–42, 56, 81, 164).

[Bar89] David A. Barrington. “Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1”. *Journal of Computer and System Sciences* 38 (1989), pp. 150–164. DOI: 10.1016/0022-0000(89)90037-8 (cit. on p. 233).

[BB14] Ämin Baumeler and Anne Broadbent. “Quantum private information retrieval has linear communication complexity”. *IACR Journal of Cryptology* 28.1 (2014), pp. 161–175. DOI: 10.1007/s00145-014-9180-2 (cit. on p. 154).

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. “Foundations of garbled circuits”. In *Proceedings of the 19th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2012, pp. 784–796. DOI: 10.1145/2382196.2382279 (cit. on p. 251).

[BCG+06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. “Secure multiparty quantum computation with (only) a strict honest majority”. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2006, pp. 249–260. DOI: 10.1109/FOCS.2006.68 (cit. on pp. 33, 42, 50, 51, 85, 86).

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. “Semi-homomorphic encryption and multiparty computation”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2011. DOI: 10.1007/978-3-642-20465-4_11 (cit. on p. 87).

[BB84] Charles H. Bennett and Gilles Brassard. “Quantum cryptography: public key distribution and coin tossing”. In. 1984, pp. 175–179. DOI: 10.1016/J.TCS.2014.05.025 (cit. on p. 5).

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. “Evaluating 2-DNF formulas on ciphertexts”. In *Proceedings of the 2nd International Conference on Theory of Cryptography (TCC)*. Springer, 2005, pp. 325–341. DOI: 10.1007/978-3-540-30576-7_18 (cit. on p. 145).

[BF10] Niek J. Bouman and Serge Fehr. “Sampling in a quantum population, and applications”. In *Advances in Cryptology – CRYPTO*. Springer, 2010, pp. 724–741. DOI: 10.1007/978-3-642-14623-7_39 (cit. on p. 134).

[Bra18] Zvika Brakerski. “Quantum FHE (almost) as secure as classical”. In *Advances in Cryptology – CRYPTO*. Springer, 2018, pp. 67–95. DOI: 10.1007/978-3-319-96878-0_3 (cit. on pp. 234, 235, 240, 241, 244, 245, 250, 251).

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. “A cryptographic test of quantumness and certifiable randomness from a single quantum device”. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 320–331. DOI: 10.1109/FOCS.2018.00038 (cit. on p. 235).

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In *Proceedings of the 3rd Conference on Innovations in Theoretical Computer Science (ITCS)*. ACM, 2012, pp. 309–325. DOI: 10.1145/2633600 (cit. on p. 145).

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. “Simpler proofs of quantumness”. In *Proceedings of the 15th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 8:1–8:14. DOI: `10.4230/LIPIcs.TQC.2020.8` (cit. on p. 235).

[BR14] Zvika Brakerski and Guy N. Rothblum. “Virtual black-box obfuscation for all circuits via generic graded encoding”. In *Proceedings of the 11th International Conference on Theory of Cryptography (TCC)*. Springer, 2014, pp. 1–25. DOI: `10.1007/978-3-642-54242-8_1` (cit. on pp. 207, 234).

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. “Efficient fully homomorphic encryption from (standard) LWE”. *SIAM Journal on Computing* 43.2 (2014), pp. 831–871. DOI: `10.1137/120868669` (cit. on pp. 145, 149, 195, 196, 230, 231).

[BY20] Zvika Brakerski and Henry Yuen. “Quantum garbled circuits”. 2020. arXiv: `2006.01085` (cit. on p. 194).

[BS17] Fernando G.S.L. Brandão and Krysta M. Svore. “Quantum speed-ups for solving semidefinite programs”. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017, pp. 415–426. DOI: `10.1109/FOCS.2017.45` (cit. on p. 5).

[BK05] Sergei Bravyi and Alexei Kitaev. “Universal quantum computation with ideal clifford gates and noisy ancillas”. *Physical Review A* 71.022316 (2005). DOI: `10.1103/PhysRevA.71.022316` (cit. on pp. 26, 134, 136).

[Bro15] Anne Broadbent. “Delegating private quantum computations”. *Canadian Journal of Physics* 93.9 (2015), pp. 941–946. DOI: `10.1139/CJP-2015-0030` (cit. on pp. 7, 145).

[Bro16] Anne Broadbent. “Popescu-Rohrlich correlations imply efficient instantaneous nonlocal quantum computation”. *Physical Review A* 94.022318 (2016). DOI: `10.1103/PhysRevA.94.022318` (cit. on p. 198).

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. “Universal blind quantum computation”. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2009, pp. 517–526. DOI: `10.1109/FOCS.2009.36` (cit. on pp. 7, 145, 146, 194).

[BGS13] Anne Broadbent, Gus Gutoski, and Douglas Stebila. “Quantum one-time programs”. In *Advances in Cryptology – CRYPTO*. Springer, 2013, pp. 344–360. DOI: `978-3-642-40084-1_20` (cit. on pp. 33, 41, 42, 45, 46, 51, 53–56, 77, 82, 167, 170, 171, 177, 195, 228, 229, 234).

[BJ15] Anne Broadbent and Stacey Jeffery. “Quantum homomorphic encryption for circuits of low T-gate complexity”. In *Advances in Cryptology – CRYPTO*. Springer, 2015, pp. 609–629. DOI: 10.1007/978-3-662-48000-7_30 (cit. on pp. 29, 146, 149, 150, 152–154, 156, 164, 193, 194).

[BJSW16] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. “Zero-knowledge proof systems for QMA”. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 31–40. DOI: 10.1109/FOCS.2016.13 (cit. on pp. 33, 53).

[BW16] Anne Broadbent and Evelyn Wainewright. “Efficient simulation for quantum message authentication”. In *Proceedings of the 9th International Conference on Information Theoretic Security (ICITS)*. Springer, 2016, pp. 72–91. DOI: 10.1007/978-3-319-49175-2_4 (cit. on pp. 34, 45, 46, 97, 161, 170, 172, 173).

[BCS12] Harry Buhrman, Matthias Christandl, and Christian Schaffner. “Complete insecurity of quantum protocols for classical two-party computation”. *Physical Review Letters* 109.160501 (2012). DOI: 10.1103/PhysRevLett.109.160501 (cit. on p. 6).

[BCWW01] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. “Quantum fingerprinting”. *Physical Review Letters* 87.167902 (2001). DOI: 10.1103/PhysRevLett.87.167902 (cit. on p. 5).

[BFSS13] Harry Buhrman, Serge Fehr, Christian Schaffner, and Florian Speelman. “The garden-hose model”. In. ACM, 2013, pp. 145–158. DOI: 10.1145/2422436.2422455 (cit. on pp. 194, 196, 198).

[CS96] A. Robert Calderbank and Peter W. Shor. “Good quantum error-correcting codes exist”. *Physical Review A* 54.1098 (1996). DOI: 10.1103/PhysRevA.54.1098 (cit. on p. 40).

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. “Obfuscating point functions with multibit output”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2008, pp. 489–508. DOI: 10.1007/978-3-540-78967-3_28 (cit. on p. 240).

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. “Obfuscation of probabilistic circuits and applications”. In *Proceedings of the 12th International Conference on Theory of Cryptography (TCC)*. Springer, 2015, pp. 468–497. DOI: 10.1007/978-3-662-46497-7_19 (cit. on p. 258).

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. "Post-quantum zero-knowledge and signatures from symmetric-key primitives". In *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1825–1842. DOI: `10.1145/3133956.3133997` (cit. on p. 141).

[CLP17] Hao Chen, Kim Laine, and Rachel Player. "Simple encrypted arithmetic library – SEAL v2.1". In *Proceedings of the 21st International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2017, pp. 3–18. DOI: `10.1007/978-3-319-70278-0_1` (cit. on p. 230).

[CEV20] Céline Chevalier, Ehsan Ebrahimi, and Quoc Huy Vu. "On the security notions for encryption in a quantum world". 2020. IACR Cryptology ePrint Archive: 2020/237 (cit. on p. 29).

[Chi05] Andrew M. Childs. "Secure assisted quantum computation". *Quantum Information & Computation* 5.6 (2005), pp. 456–466. URL: `https://dl.acm.org/doi/10.5555/2011670.2011674` (cit. on pp. 7, 145).

[CSWX14] Well Y. Chiu, Mario Szegedy, Chengu Wang, and Yixin Xu. "The garden hose complexity for the equality function". In *Proceedings of the International Algorithmic Aspects in Information and Management (AAIM)*. Springer, 2014, pp. 112–123. DOI: `10.1007/978-3-319-07956-1_11` (cit. on p. 198).

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. "Private information retrieval". *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981. DOI: `10.1145/293347.293350` (cit. on p. 145).

[Cle86] Richard Cleve. "Limits on the security of coin flips when half the processors are faulty". In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1986, pp. 364–369. DOI: `10.1145/12130.12168` (cit. on p. 90).

[CGL99] Richard Cleve, Daniel Gottesman, and Hoi-Kwong Lo. "How to share a quantum secret". *Physical Review Letters* 83.3 (1999), pp. 648–651. DOI: `10.1103/PhysRevLett.83.648` (cit. on p. 5).

[CGJV19] Andrea Coladangelo, Alex Grilo, Stacey Jeffery, and Thomas Vidick. "Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources". In *Advances in Cryptology – EUROCRYPT*. Springer, 2019, pp. 247–277. DOI: `10.1007/978-3-030-17659-4_9` (cit. on pp. 7, 145, 146).

[CDE+18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. “SPD$\mathbb{Z}_{2^k}$: efficient MPC mod 2^k for dishonest majority”. In *Advances in Cryptology – CRYPTO*. Springer, 2018, pp. 769–798. DOI: 10.1007/978-3-319-96881-0_26 (cit. on p. 87).

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. “Secure multiparty computation and secret sharing”. Cambridge University Press, 2015. DOI: 10.1017/cbo9781107337756 (cit. on pp. 85, 89).

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. “Secure multiparty quantum computation”. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2002, pp. 643–652. DOI: 10.1145/509907.510000 (cit. on p. 85).

[DS15] Wei Dai and Berk Sunar. “cuHE: a homomorphic encryption accelerator library”. In *Cryptography and Information Security in the Balkans – BalkanCryptSec*. Springer, 2015, pp. 169–186. DOI: 10.1007/978-3-319-29172-7_11 (cit. on p. 230).

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. “Multiparty computation from somewhat homomorphic encryption”. In *Advances in Cryptology – CRYPTO*. Springer, 2012, pp. 643–662. DOI: 10.1007/978-3-642-32009-5_38 (cit. on p. 87).

[DCEL09] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. “Exact and approximate unitary 2-designs and their application to fidelity estimation”. *Physical Review A* 80.012304 (2009). DOI: 10.1103/PhysRevA.80.012304 (cit. on p. 28).

[DH76] Whitfield Diffie and Martin Hellman. “New directions in cryptography”. *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638 (cit. on p. 4).

[DM17] Leo Ducas and Daniele Micciancio. “FHEW v2.0: a fully homomorphic encryption library”. 2017. URL: https://github.com/lducas/FHEW (visited on 09/11/2020) (cit. on p. 230).

[DGJ+20] Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. “Secure multi-party quantum computation with a dishonest majority”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2020, pp. 729–758. DOI: 10.1007/978-3-030-45727-3_25 (cit. on p. ix).

[DSS16] Yfke Dulek, Christian Schaffner, and Florian Speelman. “Quantum homomorphic encryption for polynomial-sized circuits”. In *Advances in Cryptology – CRYPTO*. Springer, 2016, pp. 3–32. DOI: 10.1007/978-3-662-53015-3_1 (cit. on pp. ix, 233, 234).

[DS18] Yfke Dulek and Florian Speelman. “Quantum ciphertext authentication and key recycling with the trap code”. In *Proceedings of the 13th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018, 1:1–1:17. DOI: `10.4230/LIPIcs.TQC.2018.1` (cit. on p. ix).

[DB18] Vedran Dunjko and Hans J. Briegel. “Machine learning & artificial intelligence in the quantum domain: a review of recent progress”. *Reports on Progress in Physics* 81.074001 (2018). DOI: `10.1088/1361-6633/aab406` (cit. on p. 5).

[DFPR14] Vedran Dunjko, Joseph F. Fitzsimons, Christopher Portmann, and Renato Renner. “Composable security of delegated quantum computation”. In *Advances in Cryptology – ASIACRYPT*. Springer, 2014, pp. 406–425. DOI: `10.1007/978-3-662-45608-8_22` (cit. on p. 7).

[DNS10] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. “Secure two-party quantum evaluation of unitaries against specious adversaries”. In *Advances in Cryptology – CRYPTO*. Springer, 2010, pp. 685–706. DOI: `10.1007/978-3-642-14623-7_37` (cit. on pp. 85, 87, 89, 125, 154).

[DNS12] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. “Actively secure two-party evaluation of any quantum operation”. In *Advances in Cryptology – CRYPTO*. Vol. 7417. Springer, 2012, pp. 794–811. DOI: `10.1007/978-3-642-32009-5_46` (cit. on pp. 36, 39, 85, 87, 94, 111, 116, 135, 161).

[Eke91] Artur K. Ekert. “Quantum cryptography based on Bell’s theorem”. *Physical review letters* 67.661 (1991). DOI: `10.1103/PhysRevLett.67.661` (cit. on p. 5).

[FS17] Serge Fehr and Louis Salvail. “Quantum authentication and encryption with key recycling”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2017, pp. 311–338. DOI: `10.1007/978-3-319-56617-7_12` (cit. on p. 37).

[Fil12] Maximilian Fillinger. “Lattice based cryptography and fully homomorphic encryption”. Master of Logic Project. Institute for Logic, Language and Computation (ILLC), 2012. URL: `http://homepages.cwi.nl/~schaffne/courses/reports/MaxFillinger_FHE_2012.pdf` (visited on 08/29/2020) (cit. on p. 145).

[FBS+14] K.A.G. Fisher, A. Broadbent, L.K. Shalm, Z. Yan, J. Lavoie, R. Prevedel, T. Jennewein, and K.J. Resch. “Quantum computing on encrypted data”. *Nature Communications* 5.3074 (2014). DOI: `10.1038/NComms4074` (cit. on p. 145).

[FK17] Joseph F Fitzsimons and Elham Kashefi. “Unconditionally verifiable blind quantum computation”. *Physical Review A* 96.012303 (2017). DOI: 10.1103/PhysRevA.96.012303 (cit. on p. 7).

[FHM18] Joseph F. Fitzsimons, Michal Hajdušek, and Tomoyuki Morimae. “Post hoc verification of quantum computation”. *Physical Review Letters* 120.040501 (2018). DOI: 10.1103/PhysRevLett.120.040501 (cit. on p. 7).

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. “Candidate indistinguishability obfuscation and functional encryption for all circuits”. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13 (cit. on p. 145).

[GYZ17] Sumegha Garg, Henry Yuen, and Mark Zhandry. “New security notions and feasibility results for authentication of quantum data”. In *Advances in Cryptology – CRYPTO*. Springer, 2017, pp. 342–371. DOI: 10.1007/978-3-319-63715-0_12 (cit. on pp. 33, 36, 37, 39, 42, 46, 50, 51).

[Gen09] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440 (cit. on pp. 145, 149, 246).

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. “A simple BGN-type cryptosystem from LWE”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2010, pp. 506–522. DOI: 10.1007/978-3-642-13190-5_26 (cit. on p. 145).

[GW13] Craig Gentry and Brent Waters. “Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based”. In *Advances in Cryptology – CRYPTO*. Springer, 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5 (cit. on p. 145).

[GKW15] Alexandru Gheorghiu, Elham Kashefi, and Petros Wallden. “Robustness and device independence of verifiable blind quantum computing”. *New Journal of Physics* 17.083040 (2015). DOI: 10.1088/1367-2630/17/8/083040 (cit. on p. 146).

[GKP+13a] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. “How to run turing machines on encrypted data”. In *Advances in Cryptology – CRYPTO*. Springer, 2013, pp. 536–553. DOI: 10.1007/978-3-642-40084-1_30 (cit. on p. 145).

[GKP+13b] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. "Reusable garbled circuits and succinct functional encryption". In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2013, pp. 555–564. DOI: 10.1145/2488608.2488678 (cit. on p. 145).

[GK05] Shafi Goldwasser and Yael Tauman Kalai. "On the impossibility of obfuscation with auxiliary input". In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2005, pp. 553–562. DOI: 10.1109/SFCS.2005.60 (cit. on p. 243).

[GM84] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption". *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9 (cit. on p. 145).

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. "Attribute-based encryption for circuits". In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2013, pp. 545–554. DOI: 10.1145/2488608.2488677 (cit. on p. 145).

[Got98] Daniel Gottesman. "Theory of fault-tolerant quantum computation". *Physical Review A* 57.127 (1998), pp. 127–137. DOI: 10.1103/PhysRevA.57.127 (cit. on p. 22).

[Got99] Daniel Gottesman. "The Heisenberg representation of quantum computers". In *Proceedings of the 22nd International Colloquium on Group Theoretical Methods in Physics (Group22)*. 1999. arXiv: quant-ph/9807006 (cit. on p. 23).

[Got00] Daniel Gottesman. "Theory of quantum secret sharing". *Physical Review A* 61.042311 (2000). DOI: 10.1103/PhysRevA.61.042311 (cit. on p. 5).

[GC99] Daniel Gottesman and Isaac L. Chuang. "Quantum teleportation is a universal computational primitive". *Nature* 402 (1999), pp. 390–393. DOI: 10.1038/46503 (cit. on p. 194).

[Goy18] Rishab Goyal. "Quantum multi-key homomorphic encryption for polynomial-sized circuits". 2018. IACR Cryptology ePrint Archive: 2018/443 (cit. on p. 234).

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. "Lockable obfuscation". In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017, pp. 612–621. DOI: 10.1109/FOCS.2017.62 (cit. on pp. 240, 242, 251, 252, 256).

[Gri17] Alex B. Grilo. "Relativistic verifiable delegation of quantum computation". 2017. arXiv: 1711.09585 (cit. on p. 7).

[Gri19] Alex B. Grilo. “A simple protocol for verifiable delegation of quantum computation in one round”. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl–Leibniz-Zentrum Für Informatik, 2019, 28:1–28:13. DOI: 10.4230/LIPIcs.ICALP.2019.28 (cit. on p. 146).

[Gro96] Lov K. Grover. “A fast quantum mechanical algorithm for database search”. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866 (cit. on p. 5).

[HS20] Shai Halevi and Victor Shoup. “HELib v1.1.0: an implementation of homomorphic encryption”. 2020. URL: https://github.com/shaih/HElib (visited on 09/11/2020) (cit. on p. 230).

[HM15] Masahito Hayashi and Tomoyuki Morimae. “Verifiable measurement-only blind quantum computing with stabilizer testing”. *Physical Review Letters* 115.220502 (2015). DOI: 10.1103/PhysRevLett.115.220502 (cit. on p. 7).

[HLM16] Patrick Hayden, Debbie W. Leung, and Dominic Mayers. “The universal composable security of quantum message authentication with key recyling”. 2016. arXiv: 1610.09434 (cit. on p. 37).

[Hoe63] Wassily Hoeffding. “Probability inequalities for sums of bounded random variables”. *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30. DOI: 10.2307/2282952 (cit. on p. 134).

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. “Zero-knowledge proofs from secure multiparty computation”. *SIAM Journal on Computing* 39.3 (2009), pp. 1121–1152. DOI: 10.1137/080725398 (cit. on p. 141).

[IP07] Yuval Ishai and Anat Paskin. “Evaluating branching programs on encrypted data”. In *Proceedings of the 4th International Conference on Theory of Cryptography (TCC)*. Springer, 2007, pp. 575–594. DOI: 10.1007/978-3-540-70936-7_31 (cit. on p. 151).

[Kah96] David Kahn. “The codebreakers: the comprehensive history of secret communication from ancient times to the internet”. 2nd ed. Simon and Schuster, 1996. ISBN: 0-684-83130-9 (cit. on p. 3).

[KMW17] Elham Kashefi, Luka Music, and Petros Wallden. “The quantum cut-and-choose technique and quantum two-party computation” (2017). arXiv: 1703.03754 (cit. on p. 85).

[KP17] Elham Kashefi and Anna Pappa. “Multiparty delegated quantum computing”. *Cryptography* 1.12 (2017). DOI: 10.3390/cryptography1020012 (cit. on p. 85).

[KL14] Jonathan Katz and Yehuda Lindell. “Introduction to modern cryptography”. Chapman and Hall/CRC, 2014. ISBN: 978-1-58488-551-1 (cit. on pp. 13, 17, 18, 176, 183, 209).

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. “Overdrive: making SPDZ great again” (2018), pp. 158–189. DOI: `10.1007/978-3-319-78372-7_6` (cit. on p. 87).

[Ker83] Auguste Kerckhoffs. “La cryptographie militaire”. *Journal des sciences militaires* 9 (1883), pp. 5–38. URL: `https://www.petitcolas.net/kerckhoffs/crypto_militaire_1.pdf` (visited on 08/29/2020) (cit. on p. 3).

[KP14] Hartmut Klauck and Supartha Podder. “New bounds for the garden-hose model”. In *Proceedings of the 34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS).* Schloss Dagstuhl–Leibniz-Zentrum Für Informatik, 2014, pp. 481–492. DOI: `10.4230/LIPIcs.FSTTCS.2014.481` (cit. on pp. 198, 199).

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. “Replication is not needed: single database, computationally-private information retrieval”. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS).* IEEE, 1997, pp. 364–373. DOI: `10.1109/SFCS.1997.646125` (cit. on p. 145).

[Lia13] Min Liang. “Symmetric quantum fully homomorphic encryption with perfect security”. *Quantum Information Processing* 12.12 (2013), pp. 3675 –3687. DOI: `10.1007/s11128-013-0626-5` (cit. on p. 146).

[Lia15] Min Liang. “Quantum fully homomorphic encryption scheme based on universal quantum circuit”. *Quantum Information Processing* 14.8 (2015), pp. 2749–2759. DOI: `10.1007/s11128-015-1034-9` (cit. on p. 145).

[Lo97] Hoi-Kwong Lo. “Insecurity of quantum secure computations”. *Physical Review A* 56.1154 (1997). DOI: `10.1103/PhysRevA.56.1154` (cit. on p. 6).

[LC97] Hoi-Kwong Lo and Hoi Fung Chau. “Is quantum bit commitment really possible?” *Physical Review Letters* 78.3410 (1997). DOI: `10.1103/PhysRevLett.78.3410` (cit. on p. 6).

[LC98] Hoi-Kwong Lo and Hoi Fung Chau. “Why quantum bit commitment and ideal quantum coin tossing are impossible”. *Physica D: Nonlinear Phenomena* 120 (1998), pp. 177–187. DOI: `10.1016/S0167-2789(98)00053-0` (cit. on p. 6).

[LD12] Alessio Lodola and Marco De Vivo. “The increasing role of QM/MM in drug discovery”. *Advances in Protein Chemistry and Structural Biology* 87 (2012), pp. 337–362. DOI: 10.1016/B978-0-12-398312-1.00011-1 (cit. on p. 5).

[Mah18a] Urmila Mahadev. “Classical homomorphic encryption for quantum circuits”. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 332–338. DOI: 10.1109/FOCS.2018.00039 (cit. on pp. 234, 235, 240, 241, 244, 245, 250, 251).

[Mah18b] Urmila Mahadev. “Classical verification of quantum computations”. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 259–267. DOI: 10.1109/FOCS.2018.00033 (cit. on p. 146).

[Maj20] Christian Majenz. Personal communication. Aug. 2020 (cit. on p. 25).

[Mar14] Oded Margalit. “On the riddle of coding equality function in the garden hose model”. In *Proceedings of the Information Theory and Applications Workshop (ITA)*. IEEE, 2014, pp. 1–5. DOI: 10.1109/ITA.2014.6804262 (cit. on p. 198).

[May97] Dominic Mayers. “Unconditionally secure quantum bit commitment is impossible”. *Physical Review Letters* 78.3414 (1997). DOI: 10.1103/PhysRevLett.78.3414 (cit. on p. 6).

[MDCA20] Tony Metger, Yfke Dulek, Andrea Coladangelo, and Rotem Arnon-Friedman. “Device-independent quantum key distribution from computational assumptions”. 2020. arXiv: 2010.04175 (cit. on p. x).

[Mor14] Tomoyuki Morimae. “Verification for measurement-only blind quantum computing”. *Physical review a* 89.060302 (2014). DOI: 10.1103/PhysRevA.89.060302 (cit. on p. 7).

[NC00] Michael A Nielsen and Isaac L. Chuang. “Quantum computation and quantum information”. 10th Anniversary Edition. Cambridge University Press, 2000. ISBN: 978-1107002173 (cit. on pp. 13, 40).

[OH05] Jonathan Oppenheim and Michał Horodecki. “How to reuse a one-time pad and other notes on authentication, encryption, and protection of quantum information”. *Physical Review A* 72.042309 (2005). DOI: 10.1103/PhysRevA.72.042309 (cit. on p. 37).

[OTF15] Yingkai Ouyang, Si-Hui Tan, and Joseph F. Fitzsimons. “Quantum homomorphic encryption from quantum codes”. *Physical Review A* 98 (2015). DOI: 10.1103/PhysRevA.98.042334 (cit. on p. 146).

[Pai01] Pascal Paillier. “Public-key cryptosystems based on composite degree residuosity classes”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2001, pp. 223–238. DOI: 10.1007/3-540-48910-X_16 (cit. on p. 145).

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. “Pseudorandomness of ring-LWE for any ring and modulus”. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2017, pp. 461–473. DOI: `10.1145/3055399.3055489` (cit. on p. 252).

[Por17] Christopher Portmann. “Quantum authentication with key recycling”. In *Advances in Cryptology – EUROCRYPT*. Springer, 2017, pp. 339–368. DOI: `10.1007/978-3-319-56617-7_12` (cit. on pp. 28, 33, 37, 39–43, 50, 51, 56, 60, 62, 66, 81).

[Pre97] John Preskill. “Quantum computation: lecture notes (chapter 7)”. 1997. URL: `https://www.lorentz.leidenuniv.nl/quantumcomputers/literature/preskill_7.pdf` (visited on 08/29/2020) (cit. on p. 71).

[RW75] Dijen K. Ray-Chaudhuri and Richard M. Wilson. “On t-designs”. *Osaka Journal of Mathematics* 12.3 (1975), pp. 737–744 (cit. on p. 14).

[Reg05] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2005, pp. 84–93. DOI: `10.1145/1060590.1060603` (cit. on p. 252).

[Reg10] Oded Regev. “The learning with errors problem (invited survey)”. In *Proceedings of the Annual Conference on Computational Complexity*. IEEE, 2010, pp. 191–204. DOI: `10.1109/CCC.2010.26` (cit. on p. 19).

[RUV13] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. “Classical command of quantum systems”. *Nature* 496.7446 (2013), pp. 456–460. DOI: `10.1038/nature12035` (cit. on pp. 7, 146).

[RWS+17] Markus Reiher, Nathan Wiebe, Krysta Svore, Dave Wecker, and Matthias Troyer. “Elucidating reaction mechanisms on quantum computers”. *Proceedings of the National Academy of Sciences (PNAS)* 114.29 (2017), pp. 7555–7560. DOI: `10.1073/PNAS.1619152114` (cit. on p. 5).

[RSA78] Ronald L. Rivest, Adi Shamir, and Len Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. *Communications of the ACM* 21.2 (1978), pp. 120–126. DOI: `10.1145/359340.359342` (cit. on pp. 4, 145).

[RFG12] Peter P. Rohde, Joseph F. Fitzsimons, and Alexei Gilchrist. “Quantum walks with encrypted data”. *Physical Review Letters* 109.150501 (2012). DOI: `10.1103/PhysRevLett.109.150501` (cit. on pp. 145, 146).

[SW14] Amit Sahai and Brent Waters. “How to use indistinguishability obfuscation: deniable encryption, and more”. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2014, pp. 475–484. DOI: `10.1145/2591796.2591825` (cit. on p. 145).

[SYY99] Tomas Sander, Adam Young, and Moti Yung. "Non-interactive crypto-computing for NC^1". In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1999, pp. 554–566. DOI: 10.1109/SFFCS.1999.814630 (cit. on p. 145).

[Sha48] Claude E. Shannon. "A mathematical theory of communication". *The Bell System Technical Journal* 27.4 (1948), pp. 623–656. DOI: 10.1002/j.1538-7305.1948.tb00917.x (cit. on p. 3).

[Sha49] Claude E. Shannon. "Communication theory of secrecy systems". *The Bell System Technical Journal* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x (cit. on p. 4).

[SB08] Dan Shepherd and Michael J. Bremner. "Instantaneous quantum computation". 2008. arXiv: 0809:0847 (cit. on p. 194).

[Sho94] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700 (cit. on pp. 5, 19).

[Sho96] Peter W. Shor. "Fault-tolerant quantum computation". In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1996, pp. 56–65. DOI: 10.1109/SFCS.1996.548464 (cit. on p. 71).

[SP00] Peter W. Shor and John Preskill. "Simple proof of security of the BB84 quantum key distribution protocol". *Physical Review Letters* 85 (2000), pp. 441–444. DOI: 10.1103/PhysRevLett.85.441 (cit. on p. 53).

[Spe11] Florian Speelman. "Position-based quantum cryptography and the garden-hose game". MSc thesis. University of Amsterdam, 2011. arXiv: 1210.4353 (cit. on p. 198).

[Spe16] Florian Speelman. "Instantaneous non-local computation of low T-depth quantum circuits". In *Proceedings of the 11th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, 9:1–9:24. DOI: 10.4230/LIPIcs.TQC.2016.9 (cit. on pp. 194, 198).

[Ste96] Andrew M. Steane. "Error correcting codes in quantum theory". *Physical Review Letters* 77.793 (1996). DOI: 10.1103/PhysRevLett.77.793 (cit. on p. 40).

[TKO+16] Si-Hui Tan, Joshua A Kettlewell, Yingkai Ouyang, Lin Chen, and Joseph F Fitzsimons. "A quantum approach to fully homomorphic encryption". *Scientific Reports* 6.33467 (2016). DOI: 10.1038/SRep33467 (cit. on p. 146).

[Unr15] Dominique Unruh. “Revocable quantum timed-release encryption”. *Journal of the ACM (JACM)* 62.6 (2015), pp. 1–76. DOI: 10.1145/2817206 (cit. on p. 27).

[Vai11] Vinod Vaikuntanathan. “Computing blindfolded: new developments in fully homomorphic encryption”. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2011, pp. 5–16. DOI: 10.1109/FOCS.2011.98 (cit. on p. 151).

[VFPR14] Dunjko Vedran, Joseph F. Fitzsimons, Christopher Portmann, and Renato Renner. “Composable security of delegated quantum computation”. In *Advances in Cryptology – ASIACRYPT*. Springer, 2014, pp. 406–425. DOI: 10.1007/978-3-662-45608-8_22 (cit. on p. 145).

[Wat11] John Watrous. “Theory of quantum information: lecture notes”. 2011. URL: https://cs.uwaterloo.ca/~watrous/LectureNotes.html (visited on 08/29/2020) (cit. on pp. 13, 25).

[Wee05] Hoeteck Wee. “On obfuscating point functions”. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2005, pp. 523–532. DOI: 10.1145/1060590.1060669 (cit. on p. 240).

[WC81] Mark N. Wegman and J. Lawrence Carter. “New hash functions and their use in authentication and set equality”. *Journal of Computer and System Sciences* 22.3 (1981), pp. 265–279. DOI: 10.1016/0022-0000(81)90033-7 (cit. on p. 33).

[WZ17] Daniel Wichs and Giorgos Zirdelis. “Obfuscating compute-and-compare programs under LWE”. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2017, pp. 600–611. DOI: 10.1109/FOCS.2017.61 (cit. on pp. 240, 242, 251, 252, 256).

[Wol19] Ronald de Wolf. “Quantum computing: lecture notes”. 2019. arXiv: 1907.09415 (cit. on p. 13).

[Yao82] Andrew Chi-Chih Yao. “Protocols for secure computations”. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.88 (cit. on p. 85).

[Yao86] Andrew Chi-Chih Yao. “How to generate and exchange secrets”. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25 (cit. on p. 250).

[YPF14] Li Yu, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. “Limitations on information-theoretically-secure quantum homomorphic encryption”. *Physical Review A* 90.050303 (2014). DOI: 10.1103/PhysRevA.90.050303 (cit. on pp. 6, 145).

Index

List of symbols

$\mathcal{A}$	adversary, 17
aux	auxiliary-information string, 243
b	bit or trit
C	circuit, 20; classical code, 40; Clifford element, 86
$\mathscr{C}_m$	m-qubit Clifford group, 22
$\mathbf{CC}_{f,y}$	compute-and-compare function with target value y, 247
c	circuit, 147; ciphertext, 245
c_i	block of decomposable public key, 249
$\mathcal{D}$	distinguisher (outputs a single bit), 18
$\mathfrak{D}(\cdot)$	state space, 20
d	distance of a code, 40; depth of a circuit, 246
$\mathcal{E}$	environment, 93
evk	evaluation key, 149
F	flag register, 95
G	gate, 86
$GL(n,F)$	general linear group of degree n over field F, 13
$g(sk)$	classical gadget information, 201
$\mathcal{H}$	Hilbert space, 19
$\mathfrak{I}$	ideal channel, 57; ideal protocol, 93
i, j, ℓ	indices (various meanings)
$\mathcal{J}$	interpreter, 244
K	polynomial in d and λ, 249
$\mathcal{K}$	key set, 35
k	number of parties, 85; (symmetric) key, 16
$\mathcal{L}^{\Pi}(\cdot)$	projector map, 45
M	message register, 38
$\mathcal{M}$	message generator, 18
m, n	natural number (various meanings)
O_f	(superposition) oracle for f, 26
$\mathcal{O}$	obfuscator, 243
$\mathcal{O}_Q(\cdot)$	quantum obfuscator, 254
$o_{sk,\beta}$	obfuscation of compute-and-compare function, 252
$\mathbf{P}$	polynomial time (complexity class)
P_0	identity Pauli, 21
$\mathscr{P}_m$	m-qubit Pauli group, 21
$\mathbf{P}_y$	point function with target value y, 246
pk	public key, 16
R	side-information register, 38
$\mathfrak{R}$	real channel, 57
r, s	random bit or string
$\mathcal{S}$	simulator, 18
S_n	permutation group on n elements, 22
sk	secret key, 16
T	tag register, 38
$\mathcal{T}_{\mathcal{G}}(\cdot)$	twirl over group $\mathcal{G}$, 27
$\lvert\mathsf{T}\rangle$	magic state for T, 26
$\mathrm{Tr}[\cdot]$	trace
$\mathrm{Tr}_A[\cdot]$	partial trace
t	size of tag register, 39; number of T gates in a circuit,

	204
U, V, W	unitary matrix, 20
$U^\dagger$	complex conjugate of U, 20
U^s	$U^{s_1} \otimes U^{s_2} \otimes \cdots \otimes U^{s_n}$, 20
$\mathbf{v}$	vector
$\widetilde{x}$	ciphertext for x, 150
$\widetilde{x}^{[i]}$	ciphertext for x under ith public key, 205
x_i, $x[i]$	the ith bit of a string x, 13
$\lvert x_L \rangle$	logical x, 40
$\mathbf{Z}_n$	all-zero function on n bits, 254
α	target of point function, 252
β	secret string, 252
$\Gamma_{pk}(sk)$	T-gate gadget, 202
$\gamma_{x,z}$	quantum gadget state, 201
$\gamma^{\mathsf{in}}, \gamma^{\mathsf{out}}$	first and last qubit of a gadget state, 214
γ^{mid}	middle part of a gadget state, 214
ε	small, positive real number
λ	security parameter, 16
μ^G	encoded magic state for gate G, 214
Π	projector, 20
$\overline{\Pi}$	complement $\mathbb{I} - \Pi$, 20
π	permutation, 22
ρ	density operator, 19
ρ_A	partial trace, 21
τ	fully mixed state, 20
Φ, Ψ, Λ	quantum channel, 20
Φ_C	channel induced by the circuit C, 20
$\lvert \Phi^+ \rangle$	EPR pair, 20
$\lvert \psi \rangle$	pure state, 19
$\lvert \cdot \rvert$	absolute value; Hamming weight, 13; cardinality, 13; dimension, 21
$\lVert \cdot \rVert_p$	Schatten p-norm, 24
$\lVert \cdot \rVert_{\mathrm{tr}}$	trace norm, 24
$\lVert \cdot \rVert_2$	vector 2-norm, 24; Schatten 2-norm, 24
$\lVert \cdot \rVert_\infty$	operator norm, 24
$\lVert \cdot \rVert_\diamond$	diamond norm, 26
$(\cdot)^A$	state/gate/channel in/on register A, 21
$(\cdot)^\perp$	dual of a code, 68
$(\cdot)^f$	(superposition) oracle access to f, 26
$(\cdot,\cdot)_{\mathsf{Sp}}$	symplectic product, 29
$[n]$	the set $\{1, 2, \ldots, n\}$, 13
$[[n, m, d]]$	parameters of quantum error-correcting code, 40
$\sqcup$	disjoint union, 13
$\oplus$	(bitwise) xor, 13
$\otimes$	tensor product
$\circ$	circuit composition, 14
$\diamond$	protocol composition, 14
$\leftrightarrows$	output after interaction with protocol, 14
$\leftarrow_D$	sample according to distribution D, 15
$\leftarrow_R$	uniformly random sample, 15
$\approx_\varepsilon$	statistical indistinguishability up to ε, 15; trace norm upper bounded by ε, 25; diamond norm upper bounded by ε, 26
$\overset{c}{\approx}_\varepsilon$	computational indistinguishability up to ε, 15
$\varnothing$	empty set
$\oslash$	replace with $\lvert \perp \rangle$, 161
$\lvert \perp \rangle$	reject state, 20
$+$	computational basis, 168
$\times$	Hadamard basis, 168

		see page...	extension of...	information-theoretically secure	evaluation of Clifford gates	evaluation of T gates	compact	verifiable
“trivial”	**TRIV**	150	—	✓	✓	✓	×	×
“Cliffords”	**CL**	154	—	×	✓	×	✓	×
“trap code”	**TC**	167	—	✓	×	×	×	✓
“trap code & Cliffords”	**TCL**	175	CL & TC	×	✓	×	✓	✓
“teleport”	**TP**	204	CL	×	✓	✓	✓	×
“trap code & teleport”	**TTP**	213	TP & TCL	×	✓	✓	✓	✓

Figure A: Overview of quantum homomorphic-encryption schemes discussed in Chapters 5 and 6. The table lists their abbreviations (in boldface), intended meaning (in quotes), location within this dissertation, and several key properties.

Samenvatting

Deze dissertatie verkent de mogelijkheden en onmogelijkheden om quantumberekeningen veilig te delegeren en distribueren. We bouwen concrete protocollen voor verscheidene quantumcryptografische primitieven (quantumberichtauthenticatie, meerpartijenquantumberekening en verifieerbare homomorfische quantumencryptie), maar laten ook zien dat één ander primitief onmogelijk te verwezenlijken blijkt in algemene zin (quantumobfuscatie als virtuele zwarte doos).

Voor de primitieven die we wel kunnen verwezenlijken, moeten we vaak aannames doen over de computationele vermogens van de aanvaller en diens quantumcomputer. Zulke computationele aannames zijn nodig, omdat de varianten van de gewenste primitieven informatietheoretisch onmogelijk zijn. Desondanks zijn onze protocollen informatietheoretische uitbreidingen van hun klassieke varianten: de enige computationele aannames zijn ook al nodig om de klassieke primitieven te kunnen realiseren. Het voordeel van deze aanpak is dat alle vorderingen in klassieke en post-quantumcryptografie ook direct van invloed zijn op onze protocollen.

Bij alle cryptografische primitieven die in deze dissertatie worden bestudeerd, speelt verificatie een cruciale rol. Of een client nu de uitkomst wil controleren van een berekening die hij heeft uitbesteed aan een onbetrouwbare server, of dat een speler in een meerpartijenprotocol de eerlijkheid van de andere spelers wil monitoren: de eerlijke partij heeft altijd een mechanisme nodig om zich ervan te verzekeren dat de oneerlijke partijen niet af kunnen wijken van het protocol, zonder dat dit opvalt.

In **Hoofdstuk 3** bekijken we een belangrijke bouwsteen voor verifieerbare quantumberekeningen: de quantumberichtauthenticatiecode. Hoewel die normaal gesproken wordt gebruikt om ervoor te zorgen dat een bericht niet kan worden aangepast nadat het is verstuurd, kan een berichtauthenticatiecode ook worden toegepast om één bepaalde operatie af te dwingen op het bericht: dit staat bekend als (quantum)berekening op geauthenticeerde data, ook wel (Q)CAD. We bestuderen de relatie tussen een eigenschap die (*strong*) *purity testing* heet, en het vermogen van een code om de integriteit te waarborgen van de cijfertext, in plaats van alleen van de klare tekst. Verder karakteriseren we de gevallen waarin de encryptiesleutel kan worden hergebruikt. We geven een overzicht van bestaande quantumauthenticatiecodes en bouwen twee nieuwe variaties op de zogenoemde *trap code,* een authenticatiecode die specifiek geschikt is voor QCAD. Eén van deze nieuwe variaties is *strong purity testing,* en realiseert cijfertekstauthenticatie met sleutelhergebruik. De andere variatie verschaft slechts inverse-polynomiale veiligheid, maar het versleutelen kan worden gedaan met een quantumgeheugen van constante grootte. De constructies in dit hoofdstuk zijn informatietheoretisch.

In **Hoofdstuk 4** geven we een protocol voor meerpartijenquantumberekening, waarbij een aantal spelers een gezamenlijke quantumberekening doen, terwijl ze hun invoeren geheim willen houden van de andere spelers. Voorheen waren er slechts protocollen bekend als een strikte minderheid van de spelers oneerlijk was, of als er in totaal slechts twee spelers waren. Wij generaliseren het tweespelerprotocol tot een veilig protocol voor meerpartijenquantumberekeningen voor k spelers (voor elk aantal k), en we bewijzen dat het veilig is tot $k-1$ samenspannende aanvallers. Om efficiëntie te bereiken, ontwikkelen we een nieuw publiekelijk verificatieprotocol voor de Cliffordauthenticatiecode en een testprotocol voor magischetoestandinvoeren. Ons protocol steunt op klassieke meerpartijenberekening en de bijbehorende computationele aannames.

In **Hoofdstuk 5** bestuderen we homomorfische quantumencryptie, waarbij een minder sterke client een Cliffordcircuit kan uitbesteden aan een sterkere server. Er bestaat al een simpel protocol, gebaseerd op klassieke homomorfische encryptie; wij laten zien dat het circuits geheimhoudt. Bij theoretische toepassingen van klassieke (vol)homomorfische encryptie is het vaak noodzakelijk dat de client de berekening kan verifiëren tijdens het ontsleutelen. We definiëren een nieuw primitief, “verifieerbare homomorfische quantumencryptie”, gaan nauwkeurig na welk soort compactheid in deze context kan worden verwacht, en geven twee equivalente veiligheidsdefinities: een semantische en een spelgebaseerde. We bouwen een protocol voor verifieerbare homomorfische quantumencryptie dat het mogelijk maakt om, op noninteractieve wijze, Cliffordberekeningen te delegeren en verifiëren. De verificatie is bijna helemaal klassiek; voor berekeningen die met een klassieke toestand beginnen en eindigen, is deze zelfs volledig klassiek.

In **Hoofdstuk 6** breiden we de resultaten van Hoofdstuk 5 uit naar volhomomorfishe quantumencryptie, door een procedure te ontwerpen waarmee de niet-Cliffordpoort T geëvalueerd kan worden. Met behulp van technieken van onmiddelijke nonlocale quantumberekening, construeren we een “T-poort*gadget*”, een quantumtoestand die de geheime sleutel niet onthult, maar tegelijkertijd wel de server in staat stelt om tijdens de berekening fouten te corrigeren die afhangen van die geheime sleutel. De grootte van dit *gadget* hangt af van de ruimtecomplexiteit van de ontsleutelfunctie van het onderliggende klassieke homomorfische-encryptieschema. Het resulterende protocol verschaft geheimhouding tegen aanvallen met gekozen klare teksten (CPA). We laten zien hoe dit protocol kan worden uitgebreid om ook verifieerbaarheid, in de zin van het vorige hoofdstuk, te verschaffen. Als een eerste applicatie van het verifieerbare protocol beschrijven we hoe eenmalige quantumprogramma’s kunnen worden geconstrueerd van klassieke eenmalige programma’s en volhomomorfische quantumencryptie.

In **Hoofdstuk 7** laten we zien dat het, onder een variant van de *learning-with-errors*aanname, onmogelijk is om klassieke circuits naar quantumtoestanden te obfusceren. Obfuscatie als virtuele zwarte doos is een sterk cryptografisch primitief: het versleutelt een circuit terwijl de volledige invoer-/uitvoerfunctionaliteit wordt behouden. Een opmerkelijk resultaat van Barak et al. [BGI+01] zegt dat een algemene obfusceerder die klassieke circuits naar klassieke circuits obfusceert niet kan bestaan. Een veelbelovende denkrichting, die dit onmogelijkheidsresultaat omzeilt, was om klassieke circuits naar quantumtoestanden te obfusceren, omdat die mogelijk beter in staat zijn om informatie over het geobfusceerde circuit te verbergen. Wij laten zien dat deze quantumvariant op obfuscatie als virtuele zwarte doos in het algemeen onmogelijk is voor klassieke circuits. Al doende laten we zien dat, als er klassieke hulpinvoeren aanwezig zijn die afhangen van het circuit, quantumobfuscatie als virtuele zwarte doos zelfs niet mogelijk is voor de kleine klasse van klassieke puntfuncties.

Abstract

This dissertation explores the possibilities and impossibilities of securely delegating and distributing quantum computations. We construct explicit protocols for several quantum-cryptographic primitives (quantum message authentication, multi-party quantum computation, and verifiable quantum homomorphic encryption), but show that one other primitive is impossible to achieve in general (quantum virtual black-box obfuscation).

For the primitives that we do realize, we often need to make assumptions on the computational power of the adversary's quantum computer. Such computational assumptions are necessary because the variants of the primitives that we wish to achieve are impossible information-theoretically. Nonetheless, our protocols are information-theoretic upgrades from their classical variants: the only computational assumptions are those that are already required to achieve the classical primitives. The advantage of this approach is that advances in classical and post-quantum cryptography directly influence our protocols as well.

In all of the cryptographic primitives studied in this dissertation, verification plays a crucial role. Whether a client wants to check the outcome of a computation he delegated to an untrustworthy server, or whether a player in a multi-party protocol wants to monitor the honesty of the other players in the protocol, an honest party always needs some kind of mechanism to ensure that the dishonest parties are not deviating from the protocol without being noticed.

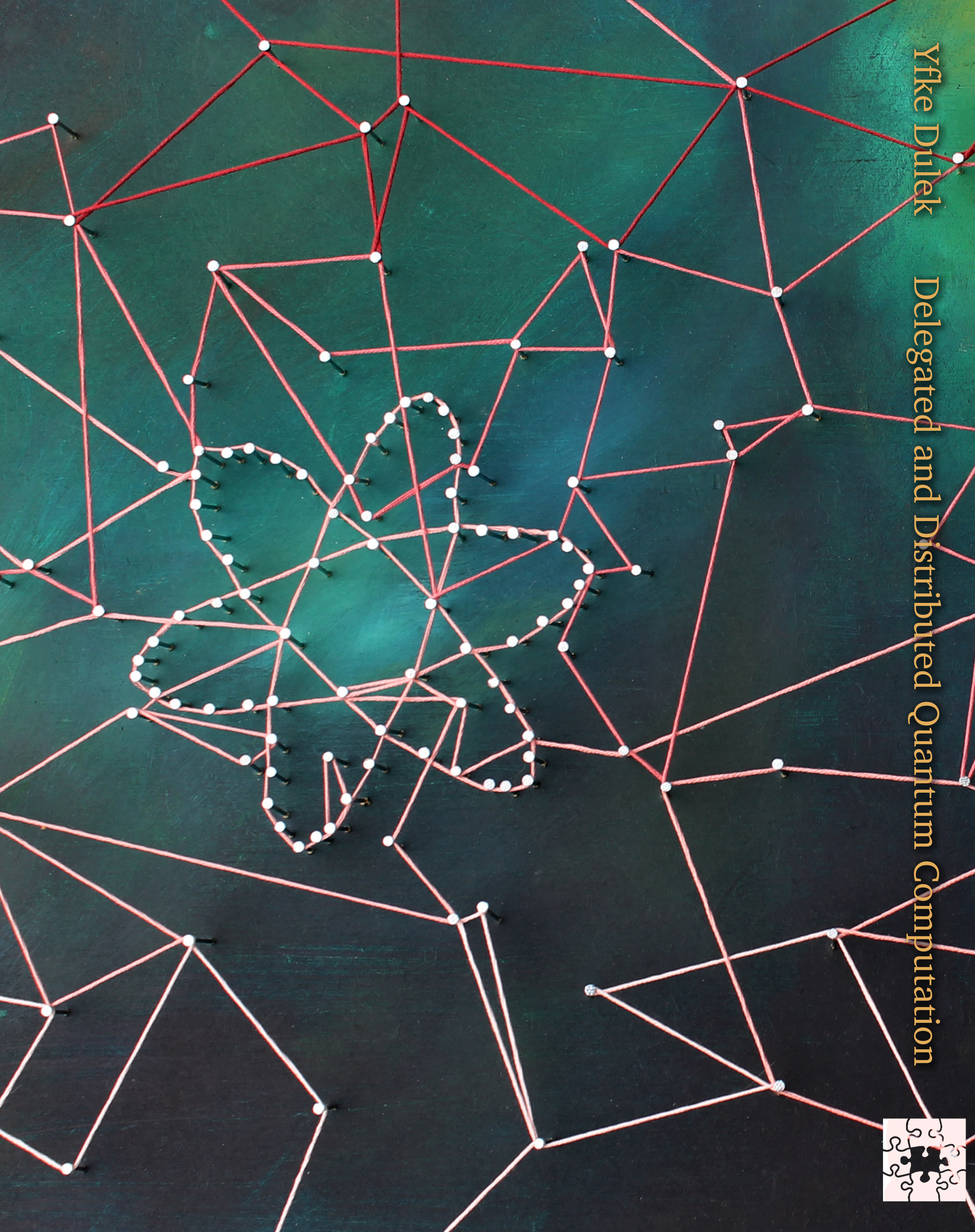
In **Chapter 3**, we consider an important building block for verifiable quantum computation: the quantum authentication code. Although traditionally used to ensure that a message cannot be altered after it is sent, authentication codes can also be used to enforce a specific operation to be applied to the message: this is known as (quantum) computing on authenticated data, or (Q)CAD. We study the relation between a property called (strong) purity testing and the ability of a code to preserve the integrity of a ciphertext rather than just the plaintext, and furthermore characterize in which cases the encryption key can be recycled. We give an overview of existing quantum authentication codes, and construct two new variations on the so-called "trap code", an authentication code that is especially suited for QCAD. One of these new variations is strong purity testing, achieving ciphertext authentication with key recycling. The other variation only achieves inverse polynomial security, but encoding can be done using only a constant-size quantum memory. The constructions in this chapter are information-theoretically secure.

In **Chapter 4**, we give a protocol for multi-party quantum computation, where a number of players perform a joint quantum computation, but want to keep their inputs private from the other players. Previously, protocols were only known if strictly less than half of the players was dishonest, or if there were only two players in total. We generalize the two-player protocol to devise a secure protocol for multi-party quantum computation for any number of players k, and prove it secure against up to $k-1$ actively colluding adversaries. To achieve efficiency, we develop a novel public verification protocol for the Clifford authentication code, and a testing protocol for magic-state inputs. Our protocol relies on classical multi-party computation and the computational assumptions that come with it.

In **Chapter 5**, we study quantum homomorphic encryption, which allows a less powerful client to outsource a Clifford circuit to a more powerful server. A simple scheme, based on classical homomorphic encryption, already exists for this task; we show that it has circuit privacy. In theoretical applications of classical (fully) homomorphic encryption, it is often necessary for the client to verify the correctness of the computation at decoding time. We define a new primitive of “verifiable quantum homomorphic encryption”, carefully consider what kind of compactness should be expected in this context, and give two equivalent definitions of security: a semantic one, and a game-based one. We construct a protocol for verifiable quantum homomorphic encryption, enabling Clifford computations to be delegated and verified in a noninteractive manner. Verification is almost entirely classical; for computations that start and end with classical states, it is completely classical.

In **Chapter 6**, we extend the results from Chapter 5 to quantum *fully* homomorphic encryption by devising a procedure to evaluate the non-Clifford gate T. Using techniques from instantaneous nonlocal quantum computation, we construct “T-gate gadgets”, quantum states that do not reveal the secret encryption key, but at the same time allow the server to correct errors that depend on that secret key during the evaluation. The size of the gadget depends on the space complexity of the decryption function of the underlying *classical* homomorphic-encryption scheme. The resulting scheme provides privacy against quantum chosen-plaintext attacks. We show how to extend it to provide verifiability, in the sense defined in the previous chapter, as well. As a first application of the verifiable scheme, we describe how to construct quantum one-time programs from classical one-time programs and verifiable quantum fully homomorphic encryption.

In **Chapter 7**, we show that, under a variant of the learning-with-errors assumption, it is impossible to obfuscate classical circuits into quantum states. Virtual black-box obfuscation is a strong cryptographic primitive: it encrypts a circuit while maintaining its full input/output functionality. A remarkable result by Barak et al. [BGI+01] shows that a general obfuscator that obfuscates classical circuits into classical circuits cannot exist. A promising direction that circumvents this impossibility result was to obfuscate classical circuits into *quantum states*, which would potentially be better capable of hiding information about the obfuscated circuit. We show that this quantum variant of virtual black-box obfuscation of classical circuits is generally impossible. On the way, we show that under the presence of dependent classical auxiliary input, even the small class of classical point functions cannot be quantum virtual black-box obfuscated.



Yfke Dulek

Delegated and Distributed Quantum Computation