Extensions Of The Garden-hose Model

MSc Thesis *(Afstudeerscriptie)*

written by

Merlijn Koek
(born December 1st, 1992 in Breda, the Netherlands)

under the supervision of **Prof Dr Harry Buhrman** and **Dr Leen Torenvliet**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
July 3rd, 2017	Prof Dr Harry Buhrman
	Yfke Dulek, MSc
	Dr Christian Schaffner
	Dr Florian Speelman
	Dr Leen Torenvliet
	Prof Dr Ronald de Wolf (Chair)

Institute for Logic, Language and Computation

Abstract

This thesis is a study of new models related to the Garden-hose model, in search of new mathematical tools for a better understanding of the Garden-hose complexity.

After providing a detailed introduction of the Garden-hose model, we introduce an extension of the model that admits multiple parties. In addition to proving several bounds on the complexity of functions in this new model, we prove the complexity of the 1-bit, k-player equality function. We sketch the applicability of the Multiparty Garden-hose model on a natural multidimensional extension of the position-verification scheme that gave rise to the original Garden-hose model.

Furthermore, we introduce the Leaky Garden-hose model, in which Alice and Bob receive more information about the water flow. We prove some results about this model, and define a family of intermediate models where all functions have a complexity between the Garden-hose complexity and the Leaky Garden-hose complexity, thus bridging the gap to the Garden-hose model.

Acknowledgement

First and foremost, I would like to thank Harry Buhrman and Leen Torenvliet for their supervision, and for showing me once again the power of creativity when combined with intelligent trial and error.

I would like to thank in particular Harry Buhrman, Christian Schaffner and Ronald de Wolf for their excellent teaching. While teaching me the necessary skills to write this thesis, they have motivated me to learn more about their fields of research.

Special thanks to Yfke Dulek, Christian Schaffner, Florian Speelman and Leen Torenvliet for our meetings, and for dedicating some of their time to proof-read key chapters.

Lastly, I would like to thank the committee for making the time to read and assess this thesis.

Motivation and Introduction

The subject of this thesis is the Garden-hose model, which has its origin in the field of position-based cryptography. The goal of this recent field of research in computer science is to use location as the key for secure communication. For example, such a cryptographic system would allow the sender of a message to be able to prove that he is at a specific geographical location. In the analysis of a position-based quantum cryptographic scheme, a family of attacks was found in Buhrman et al. [2011] that requires a number of EPR-pairs (a precious resource in quantum communication). It was found that in order to lower bound the number of EPR-pairs required in such an attack, there has to exist a function for which there are no efficient strategies to compute it in the Garden-hose model. This leads us back to the question "What is the Garden-hose model, and how are function values computed within that model?".

The Garden-hose model shares its main setting with the field of communication complexity, which is as follows. There is a publicly known Boolean function in two variables $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and two parties, Alice and Bob, who each have access to the value of one variable, x and y respectively. Now, suppose Alice and Bob are allowed to agree on a protocol before receiving their inputs x and y. How much communication between Alice and Bob is necessary before at least one of them is able to compute the value of $f(x, y)$?

The precise quantification of the necessary communication depends on the model of communication. In communication complexity, communication consists of sending bit strings back and forth, and it is quantified by the number of bits exchanged. In the Garden-hose model communication is done differently.

In the Garden-hose model, Alice and Bob have a number of water pipes in between them, with every pipe having two ends, each pipe end either being accessible only to Alice, or only to Bob. Alice and Bob are both allowed to connect pipe ends on their own side, depending on their individual input, and according to the protocol which they have agreed upon beforehand. Alice then connects a water tap to one of the pipes, as specified by the protocol, and turns on the tap. The water will then flow from the tap into the pipes, redirected by the connections made by Alice and Bob, and eventually it will spill from one of the pipes: either on Alice's side, or on Bob's side. For the protocol to correctly compute a function $f(x, y)$, the value of the function has to correspond to where the water spills.

In the Garden-hose model, communication is quantified by the minimum number of water pipes necessary to ensure that, when using an optimal protocol, $f(x, y)$ always corresponds to where the water spills. Briefly relating the Garden-hose model back to the family of attacks on position-based cryptography, the water pipes correspond to EPR-pairs that Alice and Bob (the adversaries) have to share in order to successfully execute the attack. We are interested in the relationship between the number of water pipes necessary and the corresponding function f, which is named the Garden-hose complexity of f.

Contents

Chapter 1

An Introduction to the Garden-hose Model

1.1 The Garden-hose Model

First introduced in Buhrman, Fehr, Schaffner, and Speelman [2013], the Garden-hose model is a communication complexity model, meaning that here is a publicly known Boolean function $f(x, y)$ in two variables, and there are two parties, Alice and Bob, who have access to the variables x and y respectively. The goal for Alice and Bob is to compute the value of f, on inputs x, y.

In the Garden-hose model, Alice and Bob share a finite number of water pipes located between them, which they can pairwise connect on their own side. The pipe connections on Alice's side are made depending on the input x that Alice has, and similar for Bob having input y. The output of the model is computed by opening the water tap and following the water until it spills from some pipe end at some player's side. This either happens at Alice's side when $f(x, y) = 0$, or it happens at Bob's side if $f(x, y) = 1$.

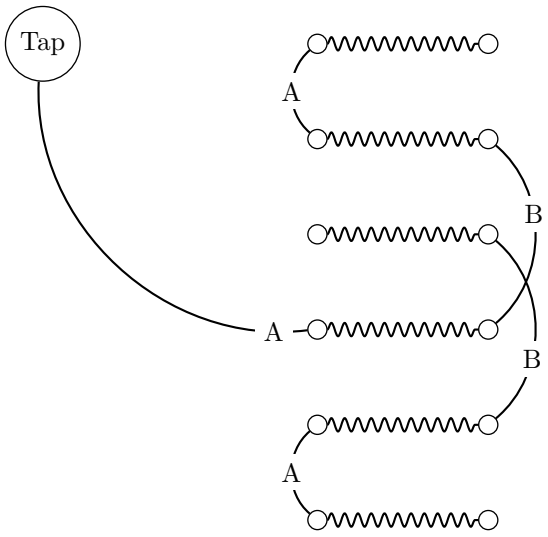

Figure 1.1: A possible combination of pipe connections.

Figure 1.1 above depicts an example where Alice and Bob share six water pipes, depicted by the waving lines. The pipe connections are depicted by the curved lines. Alice is always thought to be on the left side in these figures, and Bob on the right side. Alice has made all connections labeled A, and Bob connected his ends of the pipes with the edges labeled B. If we then follow the water from the tap, we see that in this example the water flows to the fourth pipe, then goes through pipe number two to the first pipe and ends up spilling on Bob's side.

1.1.1 Formal definition of the Garden-hose model

Formally, we define the Garden-hose model as a graph, on which certain edges are connected. This formal definition is added for clarity, even though the informal definitions of water pipes and pipe connections will be used in most parts of this thesis.

Let G_m be the graph that describes the m pipes between Alice and Bob, G_m being a balanced bipartite undirected graph on two independent sets both of size m that is 1-regular. In simpler terms, it consists of m pairs of connected vertices which share no other connections, as Figure 1.2 demonstrates for $m = 6$. We will identify the left side as Alice's side and the right side as Bob's side.

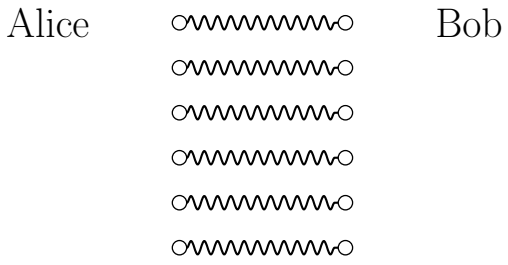

Figure 1.2: Graph G_6.

Let m be the number of pipes between Alice and Bob, and let E_A^m be the set of all non-empty* sets of pairwise pipe connections that Alice is able to make. This includes connecting an additional vertex w (the water tap) to one of the pipes. Similarly, let E_B^m be the set of all sets of pairwise pipe connections that Bob is able to make†.

Formally, we will identify all possible sets of pairwise pipe connections with the set of involutions of the symmetric group on m or $m+1$ elements. Define $E_A^m = \{\sigma \in S(m+1) : \sigma^2 = id, \sigma(w) \neq w\}$ where $S(m+1)$ is the symmetric group on $m+1$ elements, the first m elements being the m pipes between Alice and Bob, and the remaining element being the water tap w. Two pipes, e_1 and e_2, are connected on Alice's side by an involution $\pi \in E_A^m$, if $\pi(e_1) = e_2$. Note that the identity permutation is not in E_A^m, since Alice has to connect the water tap to some pipe. Similarly, define $E_B^m = \{\sigma \in S(m+1) : \sigma^2 = id\}$.

For example, labeling the three pipes simply with e_1, e_2, e_3 and the water tap with w we have:

$$E_A^3 = \{(we_1), (we_2), (we_3), (we_1)(e_2e_3), (we_2)(e_1e_3), (we_3)(e_1e_2)\}$$

Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. A Garden-hose protocol $P = (m, h_A, h_B)$ consists of the number of pipes m, corresponding to the graph G_m, and two functions h_A and h_B, where $h_A : \{0,1\}^n \to E_A^m$ describes which pipe connections are made when Alice has input $x \in \{0,1\}^n$, and similar for $h_B : \{0,1\}^n \to E_B^m$.

Consider the graph G_m, where additionally, on Alice's side the vertices are connected according to $h_A(x)$, with one of the vertices connected to an additional vertex w, the water tap, and on Bob's side the vertices are connected according to $h_B(y)$. Then compute the longest path within this graph with the restraint that one of the endpoints of this path is vertex w. This path, which we will call the ***water flow path***, corresponds to the way the water will flow when the tap is turned on. We will refer to all pipes that occur in the water flow path to be ***wet*** pipes. Similarly, we will say these pipes ***became wet on input*** (x, y). A dry pipe is a pipe that is not wet. If the other endpoint e, which is not w, of this longest path is on Alice's side, we say that the ***water***

*The water tap has to be connected to some water pipe.

†Which does contain the empty set.

spills at Alice's side (from pipe e). It is easy to see that if this is not the case, the other endpoint must be a vertex at Bob's side, and then we say the ***water spills at Bob's side***. Similarly, we call a pipe e a ***spilling pipe*** if for some input (x, y) the water spills from pipe e.

Definition 1. *A Garden-hose protocol $P = (m, h_A, h_B)$ computes a Boolean function f if, on input $x, y \in \{0,1\}^n$, and on pipe connections $h_A(x)$ and $h_B(y)$, the water spills on Alice's side if $f(x, y) = 0$ and on Bob's side if $f(x, y) = 1$.*

Notation. *We will write $GH_{spill}(h_A(x) \cup h_B(y)) = 0$ if on pipe connections $h_A(x)$ and $h_B(y)$, the water spills on Alice's side. Similarly, $GH_{spill}(h_A(x) \cup h_B(y)) = 1$ if on pipe connections $h_A(x)$ and $h_B(y)$, the water spills on Bob's side.*

It follows that a Garden-hose protocol $P = (h_A, h_B)$ computes a binary Boolean function f if $GH_{spill}(h_A(x) \cup h_B(y)) = f(x, y)$ for all $x, y \in \{0,1\}^n$.

1.1.2 The Garden-hose complexity of functions

Definition 2. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. The* Garden-hose complexity of f, $GH(f)$, *is defined to be the minimum number of pipes needed for a Garden-hose protocol to compute f.*

Consider the n-bit equality function $EQ_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $EQ_n(x, y)$ equals 1 if and only if $x = y$. The Garden-hose protocol depicted in Figure 1.3 computes EQ_n using $3n + 1$ pipes, and can be found in Speelman [2011]. Write x as a series of bits $x_1x_2 \ldots x_n$, and write y as $y_1y_2 \ldots y_n$. For each $1 \leq i \leq n$, the dashed line labeled $x_i = 0$ indicates a connection made whenever $x_i = 0$, similar for $x_i = 1$ and for the y_i.

The protocol works by checking the equality of x and y bit by bit. When $x = y$ then the water flows all the way to the bottom, always re-routed by Alice to the next darker dashed line in Figure 1.3, and ends on Bob's side, evaluating $EQ_n(x, y)$ correctly. When $x \neq y$ then at the first pair of bits $x_i \neq y_i$ Alice does not connect the pipe where Bob sent the water to, and so the water spills on Alice's side. As this protocol uses $3n+1$ pipes, we have shown the following.

Lemma 1 (Speelman [2011])**.** *Let $EQ_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be n-bit equality function. It holds that*

$$GH(EQ_n) \leq 3n + 1$$

Recent research involving scientific computation has shown that there exist more efficient protocols for the equality function: see Lemma 4 in Section 1.2.

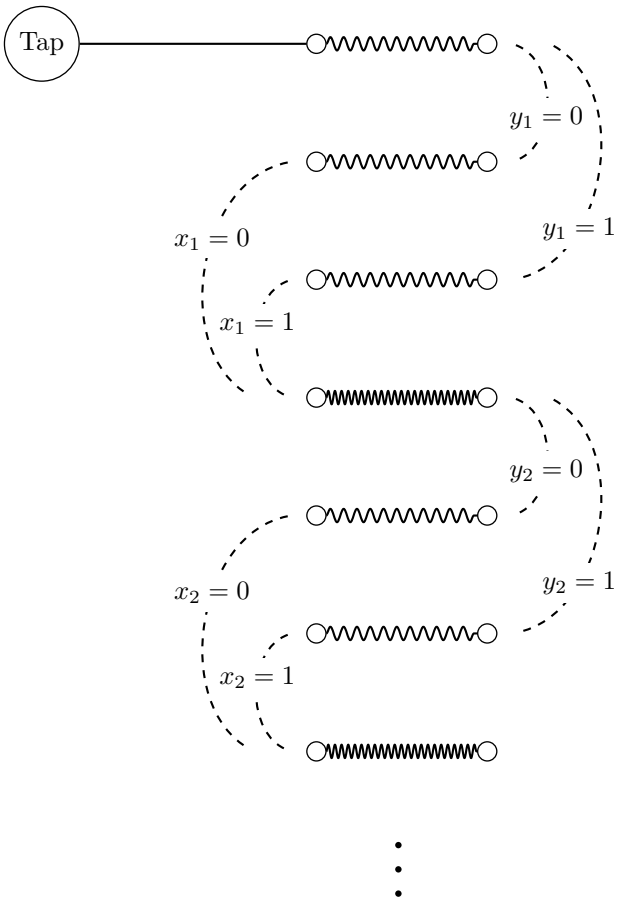

Figure 1.3: Garden-hose protocol for EQ_n.

1.2 The Configuration Matrix

Attempting to connect the Garden-hose model and classical communication complexity, we consider a configuration matrix M_m as introduced in Chiu, Szegedy, Wang, and Xu [2014]. This approach resembles the fooling set method in communication complexity (a lower bound technique, described in Kushilevitz and Nisan [1997]); We will see that the number of pipes in an optimal Garden-hose protocol computing equality is lower bounded by the smallest configuration matrix that does not contain a permutation submatrix of the required size.

In the configuration matrix M_m, each row and each column corresponds to a possible set of pipe connections that Alice and Bob respectively can make when sharing m pipes between them. Each entry (i, j) of the matrix is either 0 or 1 which is determined by taking each i and j to correspond to a unique set of

pipe connections and letting entry (i, j) equal $GH_{spill}(i \cup j)$. This means M_m is of size $|E_A^m| \times |E_B^m|$, and describes where the water comes out for all possible combinations of sets of pipe connections of Alice and Bob.

Consider M_3 and M_4. We label the water tap with 0. The pipes are labeled simply 1 to 3 (or 4) and we describe the rows and columns by a list of pipe pairings. We have left out trivial combinations where all pipes are connected as in this case the water can only come out at the other side. Moreover, the player connecting all of its pipes will know so beforehand.

$$M_3 = \begin{array}{c} \\ 01 \\ 02 \\ 03 \end{array} \begin{array}{c} \begin{array}{ccc} 12 & 13 & 23 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

$$M_4 = \begin{array}{c} \\ 01 \\ 02 \\ 03 \\ 04 \\ \mathbf{01,23} \\ \mathbf{01,24} \\ \mathbf{01,34} \\ \mathbf{02,13} \\ \mathbf{02,14} \\ 02,34 \\ \mathbf{03,12} \\ 03,14 \\ 03,24 \\ 04,12 \\ 04,13 \\ 04,23 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{12} & \mathbf{13} & \mathbf{14} & \mathbf{23} & \mathbf{24} & \mathbf{34} \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Lemma 2 (Chiu, Szegedy, Wang, and Xu [2014])**.** *$GH(EQ_n) \leq m$ if and only if M_m contains a permutation submatrix of size 2^n.*

Proof. (Chiu, Szegedy, Wang, and Xu [2014]) ($\Longrightarrow$) $GH(EQ_n) \leq m$ means EQ_n can be computed by using m pipes. Take such a protocol $P = (m, h_A, h_B)$ that computes EQ_n. In the configuration matrix M_m, the intersection of Alice's configurations $\{h_A(x)|x \in \{0,1\}^n\}$ and Bob's configurations $\{h_B(y)|y \in \{0,1\}^n\}$ is a permutation submatrix, because the entry $(h_A(x), h_B(y))$ is 1 if and only if $x = y$.

($\Longleftarrow$) We take the permutation submatrix of size 2^n. Then, we label the rows with elements in $\{0,1\}^n$ and label the columns with a permutation of $\{0,1\}^n$, such that the intersection of the row labeled with x and the column labeled y is 1 if and only if $x = y$. Finally, let $h_A(x)$ be the configuration

indicated by the row labeled with x, and $h_B(y)$ be the configuration indicated by the column labeled y. We claim $(m, h_A(x), h_B(y))$ is a protocol computing EQ_n. This is proven by the configuration matrix M_m. We can simply look up in M_m where the water spills on what combinations of sets of pipe connections, and conclude that the water spills at Alice's side if $x \neq y$, and on Bob's side if $x = y$. □

This method allows us to compute $GH(EQ_n)$ for very small n by brute-force searching through M_m for increasingly bigger m. For example, from M_4 we can learn that the maximum size of a permutation submatrix in M_4 is six. This means that using four pipes is not enough to compute equality on a set of eight elements, and we can conclude $GH(EQ_3) \geq 5$.

Leaving the proofs to be described in Chiu, Szegedy, Wang, and Xu [2014], we present two of the results that this notion of a configuration matrix has lead to.

Lemma 3 (Chiu, Szegedy, Wang, and Xu [2014])**.** *If there exist m and k such that M_m contains a permutation submatrix of size k, then it holds that*

$$GH(EQ_n) \leq \frac{m}{\log k} \cdot n + O(1)$$

Proof sketch. The idea here is to show that one can build a protocol for equality out of smaller protocols that solve equality on smaller input sizes. Specifically, it is shown that if there exists a permutation submatrix in M_m of size k, then there exists a permutation submatrix of size k^t in $M_{m \cdot t}$ for every $t \in \mathbb{N}$.

Lemma 4 (Chiu, Szegedy, Wang, and Xu [2014])**.**

$$GH(EQ_n) \leq \frac{28}{\log 3^{13}} \cdot n + O(1) \approx 1.359n + O(1)$$

1.3 Known Related Results

The field of communication complexity knows a number of 'usual suspects': functions which are extensively studied for their simple definitions while achieving high complexity. Exemplary functions are those of equality, (distributed) majority and inner product. We list definitions of these functions together with several results of bounds on their Garden-hose complexity.

Additionally, in order to gain a better insight into the context of all results in this thesis, we list a selection of general results on Garden-hose complexity. Proofs of the results below are to be found in the cited sources.

- Equality: $EQ(x, y) = 1 \iff x = y$
- Majority: $MAJ(x, y) = 1 \iff \sum_i x_i \cdot y_i \geq \lceil \frac{n}{2} \rceil$

- Inner Product: $IP(x,y) = \sum_i x_i \cdot y_i \mod 2$

Lemma 5 (Klauck and Podder [2014])**.**

$$GH(MAJ) \leq O(n \cdot \log^3 n)$$

Lemma 6 (Speelman [2011])**.**

$$GH(IP) \leq 4n + 1$$

Lemma 7 (Buhrman, Fehr, Schaffner, and Speelman [2013])**.** *Every Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has Garden-hose complexity of at most $2^n + 1$.*

Proof sketch. Use $2^n + 1$ pipes and label them each with an element in $\{0,1\}^n$, with the exception of the last pipe which we will call the reserve pipe. Let $Z(y) = \{a \in \{0,1\}^n : f(a,y) = 0\}$, group this set into pairs, and let Bob connect all pipes labeled with elements of $Z(y)$ in a pairwise manner according to this pairing. If $|Z(y)|$ is odd, connect the remaining pipe to the reserve pipe $2^n + 1$.

Although no function in particular is known to have exponential Garden-hose complexity, it does exist.

Lemma 8 (Buhrman, Fehr, Schaffner, and Speelman [2013])**.** *There exists a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with exponential Garden-hose complexity.*

However, since separating **P** and **L** has been an open problem for decades, the following lemma suggests functions with exponential Garden-hose complexity to either be very hard to find, or not be in **P**.

Lemma 9 (Buhrman, Fehr, Schaffner, and Speelman [2013])**.** *If $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is in **P** and $GH(f)$ is superpolynomial, then **P** $\neq$ **L**.*

The following trick can be used in designing Garden-hose protocols, as it allows to combine several protocols by using the spilling pipe of one protocol as a water tap for another protocol. Recall that a pipe e is called a spilling pipe if for some input (x,y) the water spills from pipe e.

Lemma 10 (One Spilling Pipe Lemma, Klauck and Podder [2014])**.** *Every Garden-hose protocol P computing a Boolean function f can be converted to a protocol P' which also computes f, but with P' constructed such that for all inputs x, y and corresponding pipe connections $h_A(x)$ and $h_B(y)$ it holds that: at Alice's side there is only one pipe from which the water can spill, and also on Bob's side there is only one pipe from which the water can spill. This means that if the water spills at Alice's side, it must spill from this one spilling pipe, and similarly on Bob's side. The number of pipes in such a protocol P' is at most three times the number of pipes in P plus one.*

Lemma 11 (Buhrman, Fehr, Schaffner, and Speelman [2013]). *Every Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has Garden-hose complexity of at most $2^{D(f)+1} - 1$, where $D(f)$ is the deterministic communication complexity of f.*

A general lower bound on the Garden-hose complexity was shown in 2014.

Lemma 12 (Klauck and Podder [2014]). *For all Boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,*

$$GH(f) \geq N(f) - 1$$

where $N(f)$ is the nondeterministic communication complexity of f.

As shown in Kushilevitz and Nisan [1997] it holds that $N(IP) \geq n+1$, which results in the following corollary.

Corollary (Klauck and Podder [2014]).

$$GH(IP) \geq n$$

1.4 The Connection with Position-based Cryptography

The goal of position-based cryptography is to use a geographical location as a key for performing cryptographic tasks. An important example is the task of position-verification, where a player, the prover, wants to prove to a group of (honest) verifiers that he is at a specific location. It was shown in Chandran et al. [2009] that if different attacking parties are allowed to collaborate, position-verification cannot be done securely in the classical setting, and later it was also shown in the quantum setting in Buhrman et al. [2011], assuming the attackers have unbounded quantum resources. Here, not being able to securely verify positions means that the attackers can prove that someone is at a specific location, without any of the attackers located there.

However, as shown in Beigi and König [2011], the general attack that breaks such a scheme uses an exponential number of EPR-pairs, which renders it practically impossible to execute. It seems that there exists a trade-off such that more classical computation done by the honest prover results in more EPR-pairs necessary to attack the scheme for the dishonest adversaries. In the introduction of the Garden-hose model in Buhrman et al. [2013], the basic scheme of position-based cryptography studied is $\mathrm{PV}^f_{\mathrm{qubit}}$, which is situated in one-dimensional space and consists of two verifiers V_0 and V_1 located far apart on a one-dimensional line, with the prover P somewhere between the verifiers. The job of the prover P is to deliver a proof of his location to the verifiers V_0 and V_1.

The scheme $\mathrm{PV}^f_{\mathrm{qubit}}$ is as follows. Verifier V_0 randomly chooses two n-bit strings $x, y \in \{0,1\}^n$ and privately sends y to V_1. Verifier V_0 also prepares an EPR-pair $\frac{1}{\sqrt{2}}(|0\rangle_V |0\rangle_P + |1\rangle_V |1\rangle_P)$. The output of a publicly known Boolean

function $f\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ then decides where V_0 sends the qubit in register V to; If $f(x,y) = 0$, it remains in register V with verifier V_0, if $f(x,y) = 1$, verifier V_0 sends the qubit in register V privately to V_1. The verifiers V_0 and V_1 then send x and y to the prover, and V_0 additionally sends the qubit in register P to the prover. Everything is sent such that it arrives at the prover at the same time. Note that this qubit in register P is entangled with the other qubit, now located at $V_{f(x,y)}$. In order to satisfyingly answer the challenge presented by the verifiers, the prover has to correctly transmit the qubit sent by V_0 to verifier $V_{f(x,y)}$. The response of the prover is verified by checking whether the qubit arrives in time at $V_{f(x,y)}$, together with performing Bell measurements of the received qubit and the other qubit sent to $V_{f(x,y)}$ by V_0, and checking whether it results in the correct outcome.

This verification uses the location of the prover to ensure an upper bound on the response time necessary to answer the challenge. It takes a certain amount of time for the input strings x and y to reach the prover, after which another stretch of time is needed to send the qubit to the correct verifier, which together upper bounds the response time of the honest prover. Any adversary trying to prove to the verifiers that he is at a specific location has to be able to answer the verifiers within this upper bound.

In an attempt to attack $\mathrm{PV}^f_{\mathrm{qubit}}$, we assume two adversaries Alice and Bob working together, trying to prove to the verifiers that they are at a location L (this location L being the location of the honest prover P), without either of the two adversaries being there. We position Alice between V_0 and L, and Bob between L and V_1. The goal for Alice and Bob is to re-route the qubit sent to L to $V_{f(x,y)}$, using the strings x, y that are also sent to L. This has to be done in the same time that the honest prover would take to complete this task. It follows from their locations that Alice is the first player to receive x and the qubit sent by V_0, and Bob is the first to receive y.

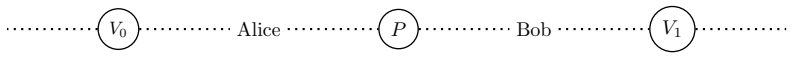

Figure 1.4: The positioning of the verifiers V_0 and V_1, the prover P and the adversaries Alice and Bob, in one-dimensional space.

What they want to achieve is a method such that if $f(x,y) = 0$, Alice holds the qubit, and if $f(x,y) = 1$, Bob holds the qubit. This has to be achieved without loss of time relative to the honest prover: if Alice and Bob first determine $f(x,y)$ in a classical communication complexity setting, their response time will be longer than the upper bound on the time the honest prover will take. If this can be done without losing time, the player in control of the qubit can then transmit the qubit to $V_{f(x,y)}$, and that qubit then can travel to the correct verifier in the same time it would have taken the honest prover to achieve. The quantum strategy which achieves this goal of attacking $\mathrm{PV}^f_{\mathrm{qubit}}$ has a one-to-one correspondence to computing f in the Garden-hose model. This correspondence was the original motivation for developing the Garden-hose model.

To execute this attack, Alice and Bob share a number of EPR-pairs (pipes in the Garden-hose model), which Alice and Bob each can 'connect' on their side by performing Bell measurements, depending on x and y respectively. (Performing such Bell measurements results in something called entanglement swapping and works such that if Alice and Bob share two EPR-pairs, and Alice performs Bell measurements on her two qubits of the two EPR-pairs, then Bob's qubits of the two EPR-pairs are entangled afterwards. A detailed introduction to quantum computation can be found in Nielsen and Chuang [2010].) In the attack, Alice teleports to Bob the qubit she receives from V_0, using the EPR-pair (pipe) that she wants to connect to the water pipe in the Garden-hose model. Then after Alice and Bob perform Bell measurements 'to connect their pipes' (which basically results in the qubit to be teleported back and forth), Alice sends x together with the outcomes of her measurements to Bob. Simultaneously, Bob sends y together with the outcomes of his measurements to Alice. If a Garden-hose protocol exists for f, the qubit ends in the hands of the correct player, and the strategy of the attack then allows Alice and Bob to also recover the qubit and send it to the correct verifier within the required time. We leave the details of how Alice and Bob recover the qubit to be explained in Buhrman et al. [2013]. Unfortunately, this attack only proves that the number of pipes necessary to compute a function f in the Garden-hose model is an upper bound on the number of EPR-pairs needed. The exact number of EPR-pairs necessary to attack $\mathrm{PV}^f_{\mathrm{qubit}}$ remains unknown and requires further investigation.

Even though the results on the Garden-hose model may not conclusively answer the question whether practically secure position-based cryptography exists, research has shown the model to have deep connections to established fields of mathematics. This, in our opinion, justifies attention to the Garden-hose model itself, as exemplified by the fact that the Garden-hose model has been linked to the long-standing question whether $\mathbf{P} = \mathbf{L}$, in Buhrman et al. [2013]. (Lemma 9 in Section 1.3)

Chapter 2

The Multiplayer Garden-hose Model

2.1 Introducing the Multiplayer Garden-hose Model

We introduce a multiplayer version of the Garden-hose model. In this model there are k players where each pair of players has a number of pipes located between them. Naturally we allow more than one edge between a pair of vertices. This forms an undirected multigraph on k vertices, where each edge corresponds to a pipe and each vertex corresponds to a player.

For each $1 \leq i \leq k$, player i is allowed to pairwise connect with hoses the edges incident to vertex i. All players make their connections based on their individual input value and the protocol that all players agreed upon. For each vertex, it holds that all edges incident to that vertex are connected according to some pairing, where we will identify such a pairing with an involution (permutation) in the following way.

Let HC_i be the set of possible combinations of the pipe connections that player i can make[†], and let E_i be the set of edges incident to player i. For $2 \leq i \leq k$, we identify HC_i with $\{\sigma \in Sym(E_i) : \sigma^2 = id\}$ where $Sym(E_i)$ is the symmetric group on E_i. Two edges e_1,$e_2 \in E_i$, are connected by player i with an involution $\pi \in HC_i$, if $\pi(e_1) = e_2$. Without loss of generality, we assume player 1 is the one to connect the water tap w to one of the edges incident to him. This means that $HC_1 = \{\sigma \in Sym(E_i \cup \{w\}) : \sigma^2 = id, \sigma(w) \neq w\}$. Note that player 1 has to connect the water tap to some pipe, so the identity permutation is not in HC_1.

Two edges e_1 and e_2 are 'connected' in the sense of the Garden-hose model if they both are incident to a vertex i such that player i has connected e_1 and e_2 with a pairing from HC_i.

[†]This corresponds to E_A^m or E_B^m in the Garden-hose model. However, different notation is chosen since the players can have a different number of pipes incident to them.

We will write $\{\{0,1\}^n\}^k$ for the collection of k input values, each input an n-bit string belonging to a single player.

Definition 3. A *Multiplayer Garden-hose protocol* $P = (M, h_1, h_2, \ldots, h_k)$ for an n-bit, k-player function $f : \{\{0,1\}^n\}^k \to \{0,1\}$ consists of the undirected multigraph M describing how many pipes any two players share, and k functions of the form $h_i : \{0,1\}^n \to HC_i$ describing which pipe connections are made when player i has input $x \in \{0,1\}^n$.

Definition 4. *The output of a protocol* P equals $i \in \{1, 2, \ldots, k\}$ when run on input $x \in \{\{0,1\}^n\}^k$ exactly when the water spills at player i when connections are made as described by the protocol.

Definition 5. *A Multiplayer Garden-hose protocol* P *computes a Boolean function* $f : \{\{0,1\}^n\}^k \to \{0,1\}$, if the set of outputs of P, when running P on all inputs in the set $\{x \in \{\{0,1\}^n\}^k \mid f(x) = 1\}$, is disjoint from the set of outputs of P, when running P on all inputs in $\{x \in \{\{0,1\}^n\}^k \mid f(x) = 0\}$.

That is, if the function has different outputs for two inputs x and y, then the water must spill at different locations. From the location of the spilling water one of the players learns the function value. This player is the one at whose side the water spills.

We define the complexity of a function in the multiplayer model, similarly to the Garden-hose complexity, as follows.

Definition 6. Let $f : \{\{0,1\}^n\}^k \to \{0,1\}$ be a Boolean function with k variables, each input variable of length n. The *Multiplayer Garden-hose Complexity of* f is the minimum number of pipes needed for a Multiplayer Garden-hose protocol to compute f. This number is denoted by $MPGH(f)$.

Note that all definitions are set up such that the following lemma holds.

Lemma 13. *For all Boolean functions* $f_n^k : \{\{0,1\}^n\}^k \to \{0,1\}$*, it holds that*

$$MPGH(f_n^2) = GH(f_n^2)$$

Proof. Let $P = (m, h_A, h_B)$ be an optimal Garden-hose protocol computing f_n^2, using m pipes. Then P can be transformed into a Multiplayer Garden-hose protocol P' computing f_n^2 by taking $P' = (M, h_1, h_2)$ where M is the multigraph with two vertices and m pipes between them, and $h_1 = h_A$, $h_2 = h_B$. □

Let us take a look at an example in Figure 2.1 for four players, Alice, Bob, Charlie, and Dick (players 1, 2, 3 and 4 respectively), where Alice controls the water tap. This is not a complete protocol computing a function, but an instance where all players have connected the edges incident to their own vertex, depending on their inputs. The example is just to see what connections are legal and how the output is determined.

Formally, if we write σ_i for the involution corresponding to the pairing of edges that player i has made, we note that $\sigma_1 = (tap, e_1)(e_3)(e_4)$, $\sigma_2 = (e_5)$,

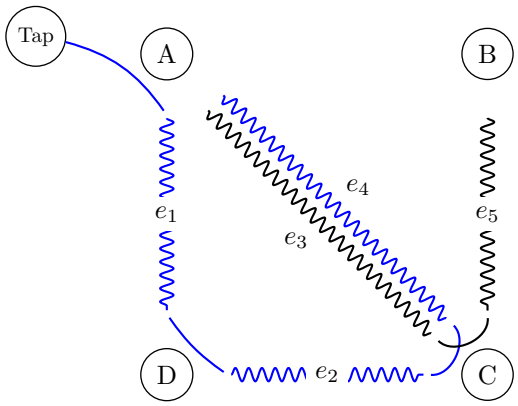

Figure 2.1: Multiparty Garden-hose example for four players. Pipes and hoses that have become wet for this configuration are drawn in blue.

$\sigma_3 = (e_2, e_4)(e_3, e_5)$ and $\sigma_4 = (e_1, e_2)$. The water flow path will be (e_1, e_2, e_4), spilling at Alice from pipe e_4. The output is 1, as Alice is player number 1.

In Section 1.3, in Lemma 10, we have seen that all Garden-hose protocols can be transformed to a protocol that has only one spilling pipe for Alice, and one for Bob. We show that this result can be translated to the Multiplayer Garden-hose model in Lemma 14 below.

Recall that a pipe e is called a spilling pipe if for some input (x, y) the water spills from pipe e. We use an analogous definition in the multiplayer model: a pipe e is called *a spilling pipe of player* P_i if for some input $(x_1, \ldots, x_k)$ the water spills from pipe e at player P_i.

Lemma 14 (Multiplayer One Spilling Pipe Lemma)**.** *Every Multiplayer Garden-hose protocol P computing a Boolean function f with k inputs can be converted to a protocol P' which also computes f, but with P' constructed such that for all inputs $x_1, x_2, \ldots, x_k$ and corresponding pipe connections $h_1(x_1), \ldots h_k(x_k)$ it holds that: for every player P_i with $1 \leq i \leq k$, there exists one pipe se^i such that if the water spills at some player P_i, it must spill from pipe se^i. The number of pipes in such a protocol P' is at most $k+1$ times the number of pipes in P plus $k-1$.*

Proof. The proof here is similar to the proof of Lemma 10 given in Klauck and Podder [2014]. Let P be a k-player protocol for computing the Boolean function $f : \{\{0,1\}^n\}^k \to \{0,1\}$. Name the players P_1 up to P_k. Let E_M be the edge set of the multigraph M corresponding to P. We construct the multigraph M' of P' by copying the structure of the original protocol $k+1$ times, such that for all pairs of players P_i and P_j with $1 \leq i < j \leq k$ it holds that if in protocol P they shared m pipes then in protocol P' they share $m \cdot (k+1)$ pipes. In addition to this, player P_1, which is the player connecting the water pipe, will share one extra pipe with every player. The total number of pipes is now $(k+1)$ times the number of pipes in P, plus $k-1$.

We will label the pipes as follows. For all pairs of players (P_i, P_j), with $1 \leq i < j \leq k$, we will label subsets of the set of pipes they share, with their copy number. Let $m(i,j)$ be the number of pipes that P_i and P_j share in protocol P. We will label $m(i,j)$ of the $m(i,j) \cdot (k+1)$ pipes between P_i and P_j in P' with the labels $(e_1^{(i,j)}, C_0), (e_2^{(i,j)}, C_0), \ldots, (e_{m(i,j)}^{(i,j)}, C_0)$, another $m(i,j)$ pipes are labeled with $(e_1^{(i,j)}, C_1), \ldots, (e_{m(i,j)}^{(i,j)}, C_1)$, et cetera, until all $m(i,j) \cdot (k+1)$ pipes are labeled.

Thus we label $m \cdot (k+1)$ pipes in P' with an index and their copy number, such that for each copy we have a one-to-one correspondence between the pipes in that copy and the pipes in M. The remaining pipes that are not yet labeled after this step are the $k-1$ pipes between P_1 and the other players. (That is, one between P_1 and P_i for every player P_i not equal to P_1, with $2 \leq i \leq k$.) Label these pipes with $e_{P_1}^i$ for all $2 \leq i \leq k$.

Next is the description of the pipe connections made by the players. Let the players connect their pipes labeled with C_0 the same as they would have connected them in P, including the water pipe, which P_1 connects to some pipe labeled C_0. Additionally, all players connect the pipes in copies C_1 to C_k as they would have connected them in P, but now player P_1 does not connect the water tap to any of the pipes.

For all $0 \leq q \leq k$ and all $1 \leq i \leq k$, define the following set $S_i^{C_q}$ of pipes in copy C_q that correspond to spilling pipes for player P_i in protocol P.

$$S_i^{C_q} = \Big\{ e \in E_i \mid e \text{ is labeled by } (e_r^{(i,j)}, C_q) \text{ for some } 1 \leq r \leq |E_i|, \\ \exists x \in \{\{0,1\}^n\}^k : \text{ the water spills from } e_r^{(i,j)} \text{ at } P_i \text{ in protocol } P \Big\}$$

Where in the second line in the definition above, $e_r^{(i,j)}$ is taken to be* the pipe that corresponds to pipe $e_r^{(i,j)}$ in M.

Now, for every $1 \leq i \leq k$, player P_i connects all the pipes $(e_r, C_0) \in S_i^{C_0}$ to their counterpart $(e_r, C_i) \in S_i^{C_i}$ in copy C_i. This means that for all $1 \leq i \leq k$, copy C_0 is connected to copy C_i only by the spilling pipes of P_i.

Let e_w denote the pipe that P_1 connects to the water pipe in copy C_0. If the water spills at P_i in protocol P, then in protocol P', the water will flow through the corresponding wet pipes labeled with C_0 and then flow through some pipe in the set $S_i^{C_0}$, corresponding to a spilling pipe for P_i in P. At that point the water will be redirected to the related pipe in $S_i^{C_i}$ and it will have the exact same flow path as in C_0 but backwards, going through the pipes in copy C_i, before spilling from the pipe in copy C_i that corresponds to pipe e_w in copy C_0.

We now know that if the water spills at player P_i in protocol P, then the water will spill at the side of P_1 from the pipe corresponding to e_w in copy number C_i. Here is where the remaining $k-1$ pipes come in.

All that is left, is for player P_1 to connect the pipe corresponding to e_w in copy number C_i to the extra pipe $e_{P_1}^i$ between P_1 and P_i. In our construction

*The protocol P' is such that in M' there are $k+1$ copies of the graph M of protocol P, where all the pipes in every copy correspond to pipes in M.

we then have $se^1 = (e_w, C_1)$ and $se^i = e^i_{P_1}$ for all $2 \leq i \leq k$, as being the spilling pipe for player P_1 and all players $P_i \neq P_1$. This protocol has only one spilling pipe per player, and uses a total number of pipes of $(k+1)$ times the number of pipes in P, plus $k-1$. □

2.2 A General Upper Bound

Similar to Lemma 7, we can show that all functions can be computed in the Multiparty Garden-hose model in exponential time.

Lemma 15. *For all Boolean functions $f : \{\{0,1\}^n\}^k \to \{0,1\}$, it holds that*

$$MPGH(f) \leq \sum_{i=1}^{k-1} 2^{ni} + \frac{1}{2} \cdot 2^{nk} < k \cdot 2^{nk}$$

Proof. Let f be a Boolean function with k n-bit variables. We construct a protocol that computes f, using $\sum_{i=1}^{k-1} 2^{ni} + \frac{1}{2} \cdot 2^{nk}$ pipes. Name the players P_1 up to P_k. Players P_1 and P_2 share 2^n pipes. Players P_2 and P_3 share $2^n \cdot 2^n$ pipes (2^n extra pipes for each pipe that P_1 and P_2 share). This continues, such that for all $1 \leq i < k$, players P_i and P_{i+1} share $(2^n)^i$ pipes. Additionally, P_1 and P_k share $\frac{1}{2} \cdot 2^{nk}$ pipes.

Let x_i denote the input of player P_i. We will label all the pipes in the following way. Label the 2^n pipes between P_1 and P_2 each with a unique input string in $\{0,1\}^n$. Label the 2^{2n} pipes between P_2 and P_3 each with a unique input string in $\{0,1\}^n \times \{0,1\}^n$. Label the 2^{3n} pipes between P_3 and P_4 each with a unique string in $\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$. Et cetera, labeling, for all $1 \leq i < k$, the 2^{ni} pipes between P_i and P_{i+1} with a unique string in $\overbrace{\{0,1\}^n \times \cdots \times \{0,1\}^n}^{i \text{ times}}$.

Having described the structure of the multigraph, we then describe the pipe connections made by the players. Player P_1 connects the water tap to the pipe labeled with his input x_1. For every $y \in \{0,1\}^n$, player P_2 connects the pipe between P_1 and P_2 that is labeled y, to the pipe labeled (y, x_2). Similarly, for every $3 \leq i < k$, and for every $y = (y_1, \ldots, y_{i-1}) \in \overbrace{\{0,1\}^n \times \cdots \times \{0,1\}^n}^{i-1 \text{ times}}$, player P_i connects the pipe labeled with y to the pipe labeled with $(y_1, y_2, \ldots, y_{i-1}, x_i)$. This means that eventually the water will flow from P_{k-1} to P_k through the pipe labeled $(x_1, x_2, \ldots, x_{k-1})$.

Player P_k then decides where the water should spill. Before connecting his pipes, Player P_k determines which set is smaller: $I_0 = \{x \in \{\{0,1\}^n\}^k : f(x) = 0\}$ or $I_1 = \{x \in \{\{0,1\}^n\}^k : f(x) = 1\}$. Let $z \in \{0,1\}$ be such that I_z is the smallest of the two sets I_0 and I_1. If I_0 and I_1 are both of size 2^{nk-1}, pick $I_z = I_0$. Note that $|I_z| \leq \frac{1}{2} \cdot 2^{nk}$.

Player P_k then connects all pipes between P_{k-1} and P_k that are labeled with a $y = y_1 y_2 \dots y_{k-1} \in \overbrace{\{0,1\}^n \times \cdots \times \{0,1\}^n}^{k-1 \text{ times}}$ such that $f(y_1, y_2, \dots, y_{k-1}, x_k) = z$, to one of the pipes between P_k and P_1.

By construction, the water will follow the pipes labeled with the inputs of the players, and it will spill at P_1 if $f(x_1, \dots, x_k) = z$ and at player P_k otherwise. Note that z can be computed before knowing the inputs. The above construction is thus a correct Multiplayer Garden-hose protocol for f, and note that the total number of pipes, excluding the pipes between P_1 and P_k, equals $\sum_{i=1}^{k-1} 2^{ni}$. Adding the pipes between P_1 and P_k, which adds another $\frac{1}{2} \cdot 2^{nk}$ pipes, finishes the proof. □

2.3 Position-based Cryptography in 3D

As we have seen in Section 1.4, the Garden-hose model models a class of attacks on a one-dimensional position-verification scheme. While for obvious practical reasons we are mainly interested in doing position-verification in 3-dimensional space, a natural follow-up of the $\text{PV}^f_{\text{qubit}}$ scheme would be to extend the scheme to k-dimensional space. In this section we will sketch the outline of such a scheme and discuss the applicability of the Multiplayer Garden-hose model.

In the k-dimensional space of this extended scheme we place $k+1$ verifiers, such that the verifiers together form a convex hull of dimension k. We require the prover to be situated within that convex hull. This is to allow verification of the location of the prover by comparing the distance upper bounds to the verifiers. The verifiers $V_0, V_1, \dots, V_k$ then send $k+1$ n-bit strings $x_0, x_1, \dots, x_k$ to the prover. Verifier V_0 again prepares an EPR-pair $\frac{1}{\sqrt{2}}(|0\rangle_V |0\rangle_P + |1\rangle_V |1\rangle_P)$, sends the qubit in register V to verifier $V_{f(x_0,x_1,\dots,x_k)}$ and sends the qubit in register P to the prover. Everything is sent such that it all simultaneously arrives at the prover. The prover then has to relay the qubit he received to $V_{f(x_0,x_1,\dots,x_k)}$, using the information that was sent to him. Here, f is a publicly known function in $k+1$ variables that can take on $k+1$ possible values.

Again, suppose there were $k+1$ adversaries, $P_0, P_1, \dots, P_k$, working together to attack this scheme, then assuming the adversaries are using a number of EPR-pairs between each pair of adversaries, such an attack corresponds to a Multiparty Garden-hose protocol for f. However, the applicability of the Multiplayer Garden-hose model as defined in this thesis is limited to some degree by two factors.

First, in the Multiplayer Garden-hose model we only consider functions of the form $f : \{\{0,1\}^n\}^k \to \{0,1\}$, whereas in the k-dimensional extension of $\text{PV}^f_{\text{qubit}}$, verifier V_0 can send his qubit in register V to any of the other verifiers. We could alter the scheme to restrict V_0 by allowing V_0 to only send the qubit to himself or V_1, which would mean that the water should only be allowed to spill at one of two players, depending on the value of the function. This is a valid solution except that we are now changing a cryptographic scheme in order

for an attack to work. Typically one designs cryptographic schemes in order to protect against possible attacks. A more elegant workaround is to redefine the Multiplayer Garden-hose model such that it deals with more general functions of the form $f : \{\{0,1\}^n\}^{k+1} \to \{0,1,2,\ldots,k_l\}$ with $1 \leq k_l \leq k$. Then if $f(x_0,x_1,\ldots,x_k) = i$ the water must spill at player P_i. In this case, the original Multiplayer Garden-hose model is simply a special case of the general model, as we can always restrict the number of players where the water can spill.

This can be done by connecting all open pipes on a player's side to extra dummy pipes. By connecting every spilling pipe of one player to an extra dummy pipe which redirects the water to another player*, we ensure that the number of players where the water can spill is reduced by one. We can do this for more than one player in order to reduce the number of spilling locations to exactly the size of the codomain of f. Note that the total number of spilling pipes, summing over all players, is upper bounded by twice the total number of pipes†. Thus we can change any Multiplayer Garden-hose protocol P which computes a Boolean function f (in the original definition), to a protocol P' which computes f such that there are only two players where the water can spill, with protocol P' using at most three‡ times the number of pipes in P.

Second, with more than two adversaries it seems possible that more complicated entangled states can be used: the attackers are not necessarily limited to using only EPR-pairs between every pair of adversaries. The Garden-hose model is already a special (although big) class of attacks, and it merely upper bounds the number of EPR-pairs needed to break the scheme $\mathrm{PV}^f_{\mathrm{qubit}}$. Since the adversaries might also use entangled states other than EPR-pairs, the Multiparty Garden-hose model represents an even smaller class of attacks on the k-dimensional extension of $\mathrm{PV}^f_{\mathrm{qubit}}$. Nevertheless, the Multiparty Garden-hose model does model a class of attacks on the natural k-dimensional extension of $\mathrm{PV}^f_{\mathrm{qubit}}$, which means it could prove a useful tool in the development of practical implementations of secure position-based cryptographic schemes in three-dimensional space.

2.4 The Multiplayer Equality Function

Let the n-bit, k-player equality function be the function $EQ^k_n : \{\{0,1\}^n\}^k \to \{0,1\}$ where EQ^k_n equals one for inputs of the form $\{x\}^k$ where $x \in \{0,1\}^n$, and zero otherwise. Before studying the complexity of the equality function in full generality in Section 2.4.5, we will take a closer look at a special case: the 1-bit equality function, starting with the double-circle protocol.

*A player that is allowed to have spilling pipes.

†Note that every pipe can have water spilling from it on two sides.

‡Using the original pipes, plus at most two dummy pipes extra for every pipe that can be a spilling pipe on two sides.

2.4.1 Example protocol: the double-circle protocol

Consider the protocol depicted in Figure 2.2. The protocol is for k players with a 1-bit input, computes EQ_1^k, and is named the double-circle protocol.

In this protocol player 1 has control over the water tap, and all the players are positioned in a circle. Every two players that are next to each other in the circle share two pipes: one pipe on the outside of the circle, and one pipe on the inside of the circle, with the exception of player 1 and player number k, who only share one pipe. The water will flow clockwise, regardless of the input.

In this protocol player 1 connects the tap to the pipe going to player 2 on the outside of the circle if his input equals 0, and he connects the water tap to the other pipe going to player 2 (on the inside of the circle) if his input equals 1. For all the other players, with the exception of player k, the protocol is to connect the two pipes on the outside of the circle if their input equals 0 and leave the remaining two pipe endings open. If their input equals 1, the protocol is to do the opposite, to connect the two pipes on the inside of the circle, and to leave the pipes on the outside unconnected.

For player k, if his input equals 0 he connects the pipe that he shares with player $k-1$ on the outside of the circle to the pipe going to player 1. If the input of player k equals 1, he will connect the pipe that he shares with player $k-1$ on the inside of the circle to the pipe leading back to player 1.

2.4.2 Analyzing the double-circle protocol

The protocol is such that if all the players have input 0, the water will flow along all the pipes on the outside of the circle and it will spill at player 1. If all players have input 1, the water will flow in a circle along the inner pipes and it will again spill at player 1, hence the name.

If not all players have the same input, the water will spill at the location of the first player i such that this player's input is not the same as the input of all the players that are closer to player 1 on the circle, counting anti-clockwise*. Note that this player is never player 1. The water only spills at player 1 if all players have the same input.

This protocol uses $2k-1$ pipes to compute EQ_1^k. It turns out a better protocol is possible, as shown in Lemma 17.

*That is, player 2 is closest to player 1, then player 3, then player 4, et cetera.

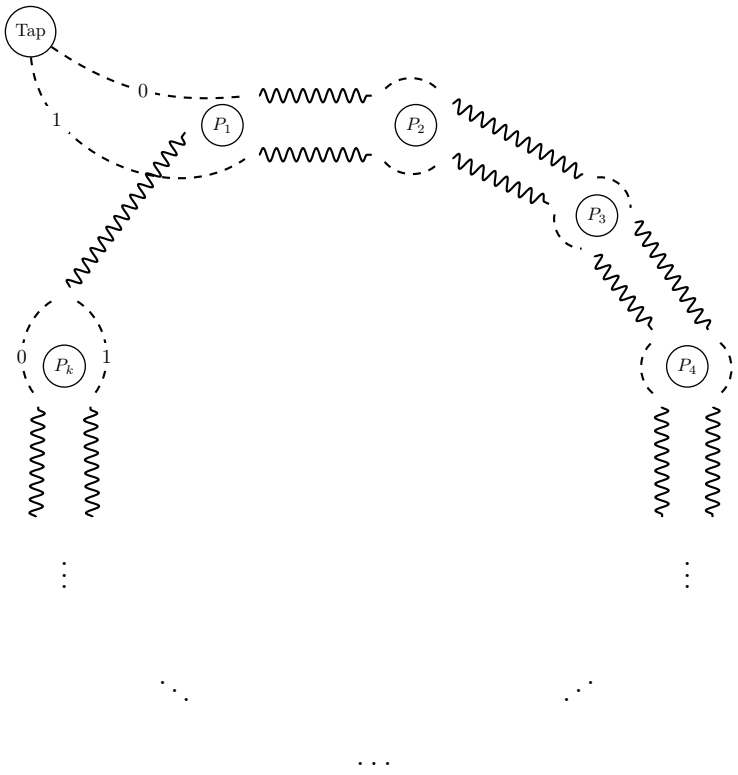

Figure 2.2: Double-circle protocol computing EQ_1^k.

2.4.3 An approach to proving optimality: a new proof technique

In an attempt to prove the optimality of the double-circle protocol, one can try to show that the following two properties must hold for every optimal protocol computing EQ_1^k.

The first property is that the number of pipes the water flows through on input $\{0\}^k$ (i.e. when all players have input zero) must be greater than or equal to $k-1$. The same is true for the number of pipes the water flows through on input $\{1\}^k$. The second property is that the set of pipes that become wet on input $\{0\}^k$ is disjoint from the set of pipes that become wet on input $\{1\}^k$, with the exception of the last pipe the water flows through.

The proof structure used here uncovers a new proof technique for Garden-hose problems: show that in any optimal protocol some input sets A and B exhibit a water flow path of significant length, and prove that the intersection of their water flow paths is small.

In the case of EQ_1^k, if these two properties hold, then at least $2k-3$ pipes are necessary to compute EQ_1^k. However, in this case the problem lies with the

second property. It turns out that these two sets* of pipes are only disjoint when the following is true: whenever a pipe s is an element of both sets, the water always flows in the same direction through pipe s. We prove this in Lemma 16.

Lemma 16. *Let I_0 be the set of pipes that become wet on input $\{0\}^k$, and let I_1 be the set of pipes that become wet on input $\{1\}^k$. The double-circle protocol is optimal up to a constant, in the set of protocols computing EQ_1^k that have the following property: whenever a pipe s is an element of both sets I_0 and I_1, the water always flows in the same direction through pipe s on every input.*

Proof. Suppose, in protocol P, that the set of pipes that become wet on input $\{0\}^k$ is *not* disjoint from the set of pipes that become wet on input $\{1\}^k$. Suppose there is a pipe that is in both sets, which is not the last pipe to become wet for either of the two inputs $\{0\}^k$ and $\{1\}^k$, *for which the water flows in the same direction on every input.*

Let e_C be the first† pipe that gets wet both on input $\{0\}^k$ and on input $\{1\}^k$. Without loss of generality we assume pipe e_C is located between player i and player j for some i and j such that $i \neq j$. Figure 2.3 depicts a schematic view of the situation.

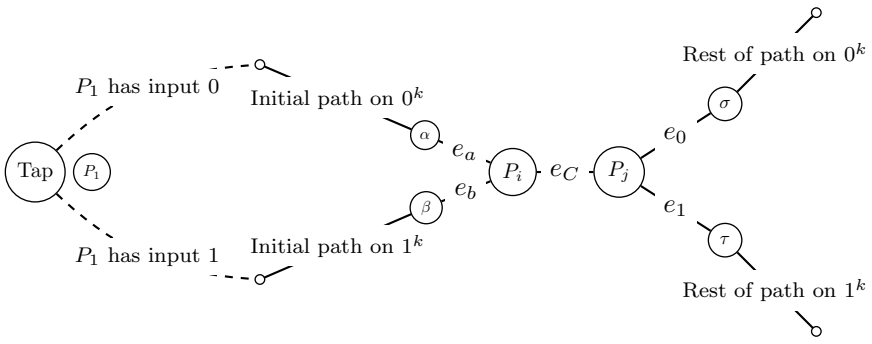

Figure 2.3: Schematic view of the water flow paths in protocol P.

Without loss of generality, player 1 has control over the water tap, and he directs the water to one of two different‡ pipes according to his input. The players (depicted by nodes in the figure) named α and β are the players incident to the pipes e_a and e_b respectively, such that the water flows through pipe e_a to player i if everyone has input 0, and the water flows through pipe e_b to player i if everyone has input 1.

*The set of pipes that become wet on input $\{0\}^k$, and the set of pipes that become wet on input $\{1\}^k$.

†Meaning that on input $\{0\}^k$, before flowing through pipe e_C, the water flows through a set of pipes E_X, and on input $\{1\}^k$, before the water flows through pipe e_C, this set E_X remains dry.

‡Suppose they were the same, then an immediate improvement can be made by having another player take control of the water tap.

The two paths coincide on edge e_C. From there, player j connects e_C to the pipe labeled e_0 if his input is 0, and he connects e_C to the pipe labeled e_1 if his input is 1. Note that the players σ and τ are possibly the same player, but we can assume that the two pipes labeled e_0 and e_1 respectively must be two different pipes; Suppose they are the same pipe, then changing pipe e_C to be between player i and σ and leaving out pipe e_0 would result in an identical protocol with fewer pipes.

We will show that in the situation depicted in Figure 2.3, the protocol is not optimal. An alternative protocol P' is depicted in Figure 2.4.

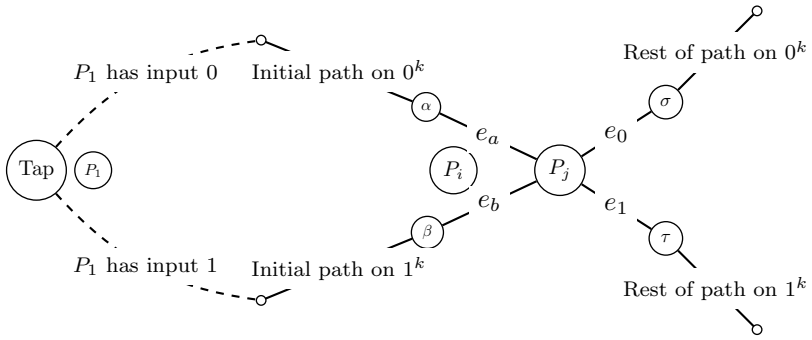

Figure 2.4: Eliminating pipe e_C from P, resulting in protocol P'.

In Figure 2.4 pipe e_C is removed, and pipes e_a and e_b are changed to be incident to P_j. In this alternative protocol P', what is different is that player P_j connects pipe e_a to the pipe labeled e_0 if he has input 0, and he connects e_b to the pipe labeled e_1 if he has input 1. It is easy to see that on inputs $\{0\}^k$ and $\{1\}^k$, protocol P' behaves the same as P.

Now suppose there exists an input $x \in \{0,1\}^k$ not equal to $\{0\}^k$ or $\{1\}^k$ for which, in protocol P, the water comes out at pipe e_a or e_b, before being channeled into pipe e_C. The water will then end up at the same pipe in protocol P' and eventually spill at the same location it would have spilled in protocol P. Hence, P' is correct and uses fewer pipes than P, contradicting the optimality of P. □

2.4.4 The exact value of $MPGH(EQ_1^k)$

From Lemma 16 we know that if there exists a better protocol for EQ_1^k, it must use a construction such that, in some pipe that occurs in both water flows, the water flows in the opposite direction for some input set. This hints towards the following protocol.

Lemma 17. *For $k \geq 3$, it holds that*

$$MPGH(EQ_1^k) \leq \left\lceil \frac{3k}{2} \right\rceil$$

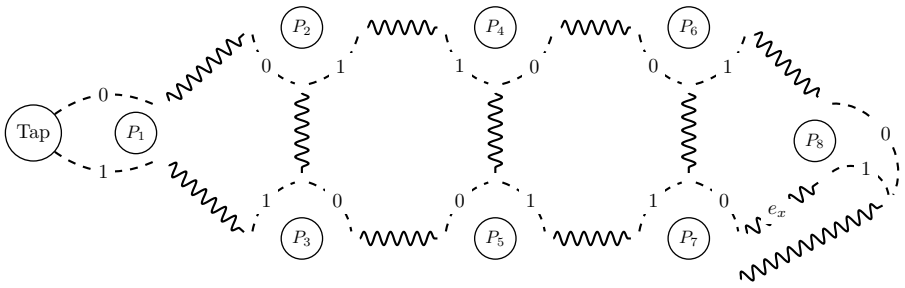

Figure 2.5: Protocol structure computing EQ_1^k if k even.

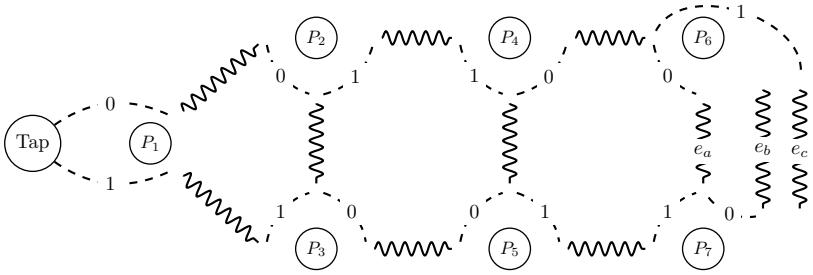

Figure 2.6: Protocol structure computing EQ_1^k if k odd.

Proof. The lemma is proven by the construction of a protocol with the correct number of pipes, separating cases for k odd and k even. (Strictly speaking, we construct a family of protocols: one for each $k \geq 3$.) Consider the protocols depicted in Figures 2.5 and 2.6. Note that by adding or removing vertical pairs of players in the middle, these protocols work for all k number of players bigger than two and even or odd respectively.

We start by proving the correctness of the protocol in Figure 2.5. It is easy to see that the water spills at P_8 if all players have the same input bit. To prove that the water does not spill at P_8 if not all players have the same input bit, consider the possible ways the water can spill at P_8.

If P_8 has input zero and the water spills at P_8, then the water must spill from the pipe between P_7 and P_8 that is left open on the side of P_8; Name this pipe e_x, as in Figure 2.5. Pipe e_x only becomes wet when P_7 has input 0. Subsequently, when P_7 has input 0, the pipe that P_7 connects to pipe e_x only becomes wet when P_6 has input 0. By repeating this argument we can follow this trail all the way back to P_1, concluding that for water to spill at P_8 when P_8 has input 0, all players must have input 0. Analogously, the same argument

holds for the case where P_8 has input 1. This proves the correctness of the protocol for even numbers of players.

Proving the correctness of the protocol in Figure 2.6 is done similarly. Here the water spills at P_6 if all players have input 0, and also when all players have input 1, as is easily checked. Label the pipes between P_6 and P_7 in Figure 2.6 with e_a, e_b and e_c. Suppose the water spills at P_6 and P_6 has input 0. Then the only possible spilling pipes are the two rightmost pipes, pipe e_b and e_c. Since P_6 has input 0, pipe e_c is not connected to any pipe and thus will not spill water. Pipe e_b will only spill water if P_7 has input 0, and backtracking from there leads to pipe e_a spilling water when all players have input 0.

Now suppose the water spills at P_6, and P_6 has input 1. The two pipes e_a and e_b between P_6 and P_7 are possible spilling pipes in this case. Pipe e_b will only spill water if both P_7 and P_6 have input 0. So if the water spills at P_6 and P_6 has input 1, it can only come out of pipe e_a, which only happens when all other players also have input 1.

In order to check the number of pipes, changing pipe e_x to be between P_8 and P_1 results in all players being incident to exactly three pipes, making it easy to see that for even k, the protocol in Figure 2.5 uses $\frac{3k}{2}$ pipes. For odd k, the protocol in Figure 2.6 uses two pipes for P_1 and three extra pipes for every other pair of players, summing up to a total of $2+\frac{3(k-1)}{2} = \frac{3k+1}{2} = k+\frac{k}{2}+\frac{1}{2} = k+\lceil\frac{k}{2}\rceil$ pipes. □

Lemma 18. *If $k \geq 3$ is even, it holds that*

$$MPGH(EQ_1^k) \geq \frac{3k}{2}$$

If $k \geq 3$ is odd, it holds that

$$MPGH(EQ_1^k) \geq k + \left\lfloor \frac{k}{2} \right\rfloor$$

Proof. Let P be an optimal protocol computing EQ_1^k. We will prove that in P, all players, with the exception of one player, are incident to three or more pipes. The proof is by contradiction, separating two cases.

First suppose that in protocol P there is a player i who is incident to only one pipe. Since there is only one pipe, it cannot be connected to another pipe. Consider the location where the water spills on input 0^k. If the water spills at player i, then the water will also spill at player i when everybody else has input 0 and player i has input 1. If it does not spill at player i, the water will again spill at the same location as where it would spill if everybody else has input 0 and player i has input 1. Basically, player i does not have any effect on the water flow. Hence P is not a protocol computing EQ_1^k and we see that all players are incident to at least two pipes.

Suppose there are two players, A and B, that are incident to exactly two pipes. Without loss of generality, assume that on input 0^k the water will flow

first to player A and then to player B. (If the water flow does not reach one of the two at all, then changing this player's input to 1 will not change the spilling location and P would not be computing EQ_1^k.) This means that player A connects her two pipes on input 0, and disconnects them on input 1. (Player A must do something different for different inputs, otherwise the water flow on input 0^k is the same as when everybody else has input 0 and player A has input 1.)

Now consider the water flow on input 1^k. If the water flows towards player A first, then the water will never reach player B, as player A has disconnected her two pipes on input 1. So the water will flow through the pipes of player B first. This means that the pipes of player B must be connected on input 1, and be disconnected on input 0, so the water spills at player B on input 0^k, otherwise the water never reaches player A.

Because player B disconnects his pipes on input 0, and since the water will flow through his pipes when everyone has input 1, the water also must* spill at player B when everybody else has input 1 and player B has input 0. This contradicts the assumption of P computing EQ_1^k, as the water spills at player B if all players have input 1, but the water spills again at player B if the only player with input 0 is player B.

Note that if k is even, it can't be the case that $k-1$ players are incident to exactly three pipes and one player is incident to exactly two pipes: the sum of the degrees (number of incident pipes) of all vertices (players) would be an odd number. This means the sum of the degrees has to increase by at least one. Dividing the sum of the degrees by two, we conclude there must be at least $\frac{3k}{2}$ pipes if k is even. If k is odd, the total number of pipes equals $\frac{3(k-1)+2}{2} = \frac{3k}{2} - \frac{1}{2} = k + \left\lfloor \frac{k}{2} \right\rfloor$. □

Combining Lemmas 17 and 18 results in the following theorem.

Theorem 1. *For $k \geq 3$, it holds that*

$$\left\lfloor \frac{3k}{2} \right\rfloor \leq MPGH(EQ_1^k) \leq \left\lceil \frac{3k}{2} \right\rceil$$

2.4.5 The complexity of n-bit, k-player equality

Using the result of Theorem 1, we are able to start to answer a bigger question: "How many pipes are needed to compute n-bit, k-player equality?".

An upper bound

An upper bound can be achieved by stringing together copies of the protocol in Lemma 17. Figure 2.7 shows how this can be done for odd k. In the upper

*This is only true because with just two pipes incident to him, player B has no influence on the water flow before it flows through a pipe incident to him, ensuring that the water flow will reach player B with the same preceding path when all players except B have input 1, independent of the input of player B.

copy, all players check their first input bit: all connections in the upper copy are made according to all players' first input bit. An extra pipe e_X is added between player $k-1$ and player 1. For input $i \in \{0,1\}$, player $k-1$ additionally connects pipe e_b to e_X if $i = 0$ and pipe e_a to X if $i = 1$. Player 1 now uses the water that spills from pipe e_X as a water tap for use in the second copy. In the lower copy, all players check their second input bit.

Note that this protocol now means that all players except players 1, $k-1$ and k are incident to exactly six pipes. By repeating this process of stringing together copies, such that in each copy another input bit location is checked to be equal for all players, we arrive at a protocol for n-bit, k-player equality that uses $n \cdot \lceil \frac{3k}{2} \rceil + (n-1)$ pipes.

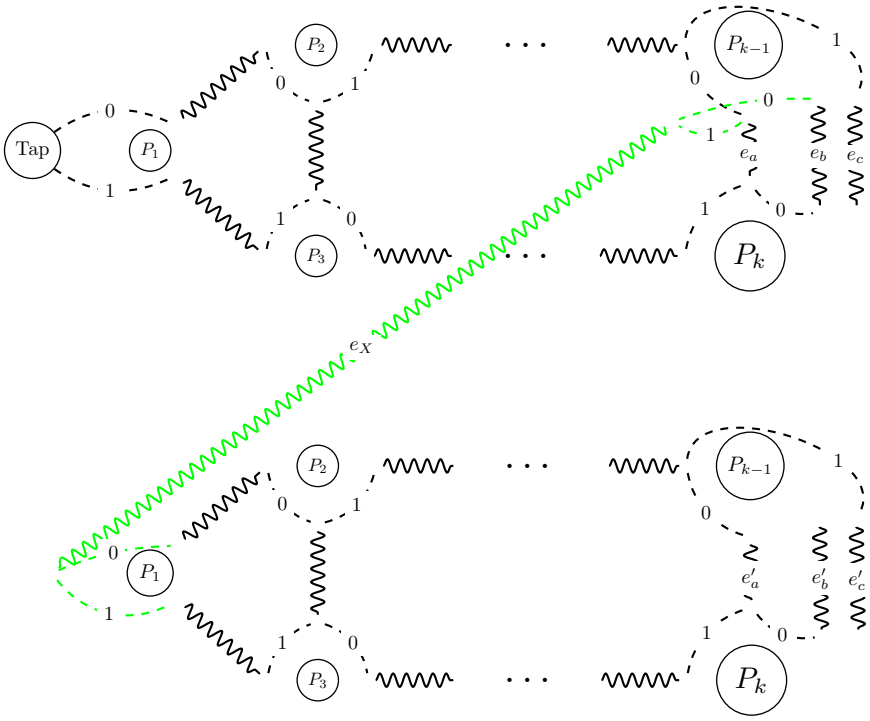

Figure 2.7: Protocol structure computing EQ_2^k if k odd.

Lemma 19. *For $k \geq 3$ it holds that*

$$MPGH(EQ_n^k) \leq n \cdot \left\lceil \frac{3k}{2} \right\rceil + (n-1)$$

Proof. For odd k, expand the protocol that is depicted in Figure 2.7 by connecting n copies of the protocol in Lemma 17 using connective pipes like pipe

e_X in Figure 2.7. Similar to the correctness proof in Lemma 17, we can show that the water only spills at player $k-1$ if all players have the same input. For each $1 \leq l \leq (n-1)$, copy C_l checks whether all players agree on input bit l, and if that is the case, the water gets transmitted to copy $l+1$. Note that the water spills at player $k-1$ if all players have the same input bit for each bit location $1 \leq l \leq (n-1)$.

To complete the correctness proof, we claim that the water only spills at player $k-1$, and, specifically, only from a pipe in the n-th copy, if all players agree on their entire input. This can be seen by noting that the water only flows towards player $k-1$ at copy number l if: all players agree on their input on all bit locations lower than l, and all players except possibly player $k-1$ and player k agree on input bit i. Three situations can occur if the water flows towards player $k-1$ at copy number l: the water spills before it reaches player $k-1$; Player $k-1$ disagrees with some player on input bit location l, and the water spills somewhere else; Or player $k-1$ agrees with all other players on input bit l, and the water gets redirected to copy $l+1$.

For even k, we can analogously combine copies of the protocol for even k in Lemma 17. On input 0, player k then connects the pipe where the water would spill from if all players have input 0 to pipe e_X, connecting two copies, and the water tap is again replaced by pipe e_X. Checking the correctness of this new protocol is done in the same way. □

A lower bound

Although finding the exact value of $MPGH(EQ_n^k)$ is likely to be difficult, as is shown by the complex nature of $MPGH(EQ_n^2) = GH(EQ)$, we can find a loose lower bound by an analysis similar to the one used in Lemma 18.

Lemma 20. *Let $T(m)$ be the number of different combinations of pipe connections a player can make when incident to m pipes, and let $L(m)$ be the least $m \in \mathbb{N}$ such that $T(m) \geq 2^n$. Then it holds that*

$$MPGH(EQ_n^k) \geq \frac{1}{2} \cdot k \cdot L(n)$$

Proof. Let P be a protocol computing equality, and suppose there exist an $x, y \in \{0,1\}^n$ such that there is a player $1 \leq i \leq k$ that makes the same combination of pipe connections on input x as he does on input y, according to protocol P. Then if all players have input x, the water must spill somewhere else than in the case where player i has input y but all other players have input x. But since the pipe connections are the same in this case, this contradicts the assumption that P is a protocol computing equality.

This means that all players must be allowed to have a unique combination of pipe connections for each input in $\{0,1\}^n$, which asserts each player to be incident to at least $L(n)$ pipes. □

The number of different combinations of pipe connections on n pipes is equal to the number involutions. This integer sequence $T(n)$ for $n \in \mathbb{N}$ is known as the 'telephone numbers', and can be found in Sloane and Plouffe [1995] and at https://oeis.org/A000085.

The table below shows the first fifteen telephone numbers $T(n)$ and the values of $L(n)$ for $1 \leq n \leq 15$.

n	$T(n)$	2^n	$L(n)$
1	1	2	2
2	2	4	3
3	4	8	4
4	10	16	5
5	26	32	6
6	76	64	6
7	232	128	7
8	764	256	8
9	2620	512	8
10	9496	1024	9
11	35696	2048	9
12	140152	4096	10
13	568504	8192	10
14	2390480	16384	11
15	10349536	32768	11

Comparing the upper and lower bounds of respectively Lemma 19 and Lemma 20 for $1 \leq n \leq 15$, we see a slowly widening gap in the table below.

n	lower bound $\frac{1}{2} \cdot k \cdot L(n)$	upper bound $n \cdot \left\lceil \frac{3k}{2} \right\rceil + (n-1)$
1	k	$1.5k$
2	$1.5k$	$3k+1$
3	$2k$	$4.5k+2$
4	$2.5k$	$6k+3$
5	$3k$	$7.5k+4$
6	$3k$	$9k+5$
7	$3.5k$	$10.5k+6$
8	$4k$	$12k+7$
9	$4k$	$13.5k+8$
10	$4.5k$	$15k+9$
11	$4.5k$	$16.5k+10$
12	$5k$	$18k+11$
13	$5k$	$19.5k+12$
14	$5.5k$	$21k+13$
15	$5.5k$	$22.5k+14$

2.4.6 Comparison to 2-player equality

By using the One Spilling Pipe Lemma (Lemma 10 from Section 1.3) we can relate $MPGH(EQ_n^k)$ to $GH(EQ_n)$, shown in Lemma 21 below.

Lemma 21. *For all* $k \in \mathbb{N}$

$$MPGH(EQ_n^k) \leq (k-1) \cdot (3 \cdot GH(EQ_n) + 1)$$

Proof. Let P be an optimal protocol computing $MPGH(EQ_n^2)$. Let P' be a protocol computing $MPGH(EQ_n^2)$ with only two spilling pipes as provided by Lemma 10, meaning that there is a pipe a such that if $x \neq y$ then the water spills at Alice's side from her end of pipe a, and there is a pipe b such that if $x = y$ then the water spills at Bob's side from his end of pipe b. Note that protocol P' uses at most $3 \cdot GH(EQ_n) + 1$ pipes.

We construct a multiplayer protocol of the requested size such that the water spills at player k if and only if all players have the same input. This is done by setting up a string of copies of protocol P' between the players in a linear pairwise fashion as shown in Figure 2.8.

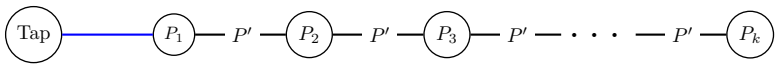

Figure 2.8: Computing EQ_n^k using copies of an optimal protocol for EQ_n^2.

Players P_1 and P_2 run protocol P', player P_2 then uses his only spilling pipe between P_1 and P_2 as a water tap for another copy op P' between P_2 and P_3. Similarly, each player other than P_1 and P_k uses their spilling pipe from a copy of the protocol P' on the left as a water tap for the copy of P' on their right.

In this manner, first P_1 and P_2 check if they have the same input. If not, then the water spills at player P_1. If so, then the next couple P_2 and P_3 check their input for equality. This continues in the same way up to player P_k.

If all players have the same input, the water spills at player P_k. If not, then the water spills at the first player P_i* such that the input of P_{i+1} is not equal to the input of player P_i. This uses a number of pipes exactly $k-1$ times the size of P'. □

Although no exact value for $GH(EQ_n) = MPGH(EQ_n^2)$ is known, several bounds exist. The best current upper bound, from Lemma 4 in Section 1.2, shows that $GH(EQ_n) \leq \frac{28}{\log 3^{13}} \cdot n + O(1) \approx 1.359n + O(1)$. Together with Lemma 21 this results in the following upper bound on $MPGH(EQ_n^k)$.

Lemma 22. *For all* $k \in \mathbb{N}$

$$MPGH(EQ_n^k) \leq (k-1) \cdot \left(3 \cdot \frac{28}{\log 3^{13}} \cdot n + O(1)\right)$$

*That is, the player with the lowest index number.

Recall Lemma 19 from Section 2.4.5, as repeated below, and note that it implies that $MPGH(EQ_n^k) \leq n \cdot \left(\frac{3k+3}{2}\right)$ for all $k \geq 3$.

Lemma 19. *For $k \geq 3$ it holds that*

$$MPGH(EQ_n^k) \leq n \cdot \left\lceil \frac{3k}{2} \right\rceil + (n-1)$$

Comparing this with Lemma 22, we see that Lemma 19 provides the current best upper bound on the n-bit, k-player equality function.

Chapter 3

Leaking Pipes and Time Bounds

In the first part of this chapter, we introduce a stronger version of the Garden-hose model: the Leaky Garden-hose model. The second part of this chapter revisits a model in which the time it takes for the water to spill is limited, a model which was first introduced in Klauck and Podder [2014].

3.1 Leaky Garden-hose Model

We introduce the Leaky Garden-hose model. The Leaky Garden-hose model has the same setting as the original Garden-hose model, but in this model, the pipes are leaking; Alice and Bob are allowed to measure which pipes get wet when running the protocol on input (x, y), which is coded in an m-bit string $LGH_P(x, y)$, where m is the number of pipes used in the protocol. Each bit at location $1 \leq i \leq m$ of $LGH_P(x, y)$ corresponds to whether pipe i became wet or not. The m-bit string $LGH_P(x, y)$ is called *the auxiliary output*, as it will be used to help compute the function value.

We say that the *Leaky Garden-hose protocol P computes a function f* if, after the water spills somewhere, and after Alice and Bob learn $LGH_P(x, y)$, it holds that for all inputs x, y, at least one player knows with certainty the value of $f(x, y)$.

In order to clarify the model, we will explicitly state the chronological structure. First, Alice and Bob agree on a protocol $P = (m, f_A, f_B)$. Then Alice and Bob receive their respective inputs x and y, and then connect their pipes according to $f_A(x)$ and $f_B(y)$ respectively. After that, the water tap is turned on and the water spills at either Alice or Bob. Then Alice and Bob measure $LGH_P(x, y)$ (thus learning which pipes became wet), after which, as the final step, at least one of them is able to compute $f(x, y)$. This definition is in line with the Multiplayer Garden-hose model, where after the water spills at some player i, this player i is able to compute f. In both models at least one player

learns the value of the function, while other players (possibly) do not.

As before, we define the Leaky Garden-hose complexity of a function as follows, using the number of pipes as a complexity measure. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function.

Definition 7. The *Leaky Garden-hose complexity* of f, $LGH(f)$ is the minimum number of pipes needed for a Leaky Garden-hose protocol to compute f.

Note that the Leaky Garden-hose model has no natural correspondence to the position-verification scheme in Section 1.4. After Alice and Bob have received the input value of the other player in scheme $\text{PV}^f_{\text{qubit}}$, they both don't know which EPR-pairs were used (i.e. which pipes became wet). Moreover, in the Garden-hose model the water has to spill at the correct player in order to compute a function [*], as opposed to Alice and Bob learning the value of the function after executing the protocol. Nevertheless, even though a direct correspondence is lost here, we believe that studying related models can improve our understanding of the underlying mathematics of the Garden-hose model.

3.1.1 Complexity upper bounds

Lemma 23. *For all Boolean functions* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$LGH(f) \leq GH(f)$$

Proof. Let $P = (m, f_A, f_B)$ be an optimal Garden-hose protocol that computes f. We claim that protocol P is also a Leaky Garden-hose protocol that computes f. This holds because $f(x,y)$ can be computed using the parity of the Hamming weight of $LGH_P(x,y)$. The water spills at Bob's side when the number of wet pipes is odd, and the water spills at Alice's side when the number of wet pipes is even. □

In contrast to the existence of functions with exponential complexity in the Garden-hose model, all functions have a complexity in $O(n)$ in the Leaky Garden-hose model.

Lemma 24. *For all Boolean functions* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$LGH(f) \leq 4n - 2$$

Proof. The protocol that shows this to be true can be seen in Figure 3.1: Bob always redirects the water to the next set of two pipes, and except for the first two pipes, Alice connects her pipes in parallel or crosswise, depending on the values of $x = x_1, x_2, x_3, \ldots, x_n$. That is, labeling the pipes from top to bottom

[*] Corresponding to sending the qubit to the correct verifier in $\text{PV}^f_{\text{qubit}}$.

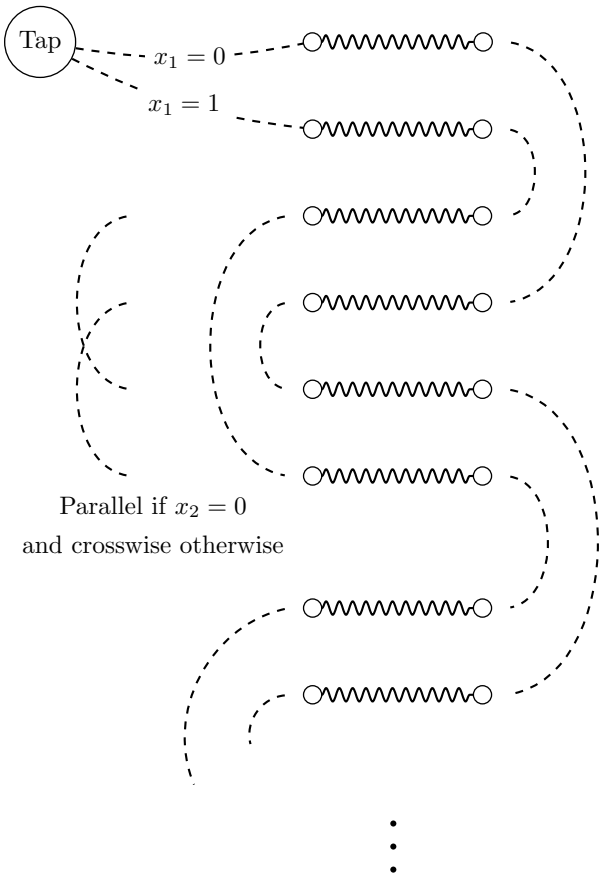

Figure 3.1: Leaky Garden-hose protocol, communicating x to Bob in $4n - 2$ pipes

with a number between one and $4n - 2$, the protocol describes the following connections. Alice connects the water tap to pipe one if $x_1 = 0$, and Alice connects the water tap to pipe two if $x_1 = 1$. Also, for all $i \in \mathbb{N}$ with $i \leq n - 1$ it holds that: if $x_{i+1} = 0$ then Alice connects pipe $4i - 1$ to pipe $4i + 2$, and pipe $4i$ to pipe $4i + 1$; If $x_{i+1} = 1$ then Alice connects pipe $4i - 1$ to pipe $4i + 1$, and pipe $4i$ to pipe $4i + 2$.

After the water flows, Bob learns which pipes get wet, and then Bob checks for all pipes whether the wet pipes must have been connected in parallel or crosswise on Alice's side. Since these connections have a one-to-one correspondence to x, Bob now knows both x and y, and he can compute $f(x, y)$. This protocol uses $4n - 2$ pipes: two for the first bit of x and four for every next bit of x, thus proving Lemma 24. □

3.1.2 Two example protocols for the equality function

Recall that because of Lemma 23, we know that for all Boolean functions f it holds that $LGH(f) \leq GH(f)$. Towards a better understanding of the complexity of functions in the Leaky Garden-hose model, we investigate $LGH(EQ_1)$ and $LGH(EQ_2)$.

A simple protocol in Figure 3.2 shows that $LGH(EQ_1) = 1$. Here, Alice only connects the water pipe if $x = 1$. When the water flows, Bob sees whether the water pipe becomes wet or not, learns Alice's bit x from this, and is able to compute $EQ_1(x, y)$ using one pipe. However trivial this may be, this shows that for some $n \in \mathbb{N}$ it holds that $LGH(EQ_n) < GH(EQ_n)$, as it is easy to see that it is impossible to compute equality using only one pipe in the Garden-hose model.

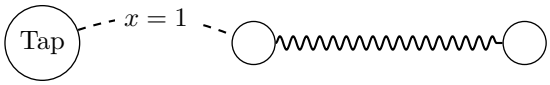

Figure 3.2: Leaky Garden-hose protocol for 1-bit equality using one pipe.

Figure 3.3 shows a faulty protocol for 2-bit equality. Checking the correctness of such a protocol can be done using a table such as in Figure 3.4. A crucial step in this process is to check the following: in the case that there are two inputs (x, y) and (x, y') with $y \neq y'$ such that $LGH_P(x, y) = LGH_P(x, y')$, then Bob should always be the one that has to know $f(x, y)$ afterwards. Similarly, this also has to hold for two inputs (x, y) and (x', y) with $x \neq x'$: in this case Alice should always have learned $f(x, y)$. However, the protocol in 3.3 is incorrect for another reason.

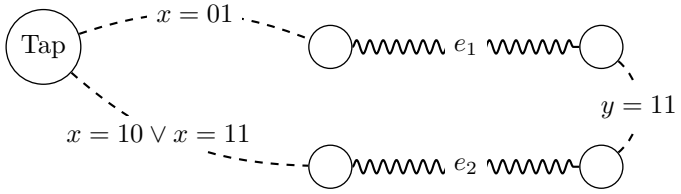

Figure 3.3: Incorrect Leaky Garden-hose protocol for 2-bit equality.

x	y	Wet pipes	Who learns $EQ(x,y)$
00	00	None	Bob
01	00	e_1	Bob
10	00	e_2	Bob
11	00	e_2	Bob
00	01	None	Bob
01	01	e_1	Bob
10	01	e_2	Bob
11	01	e_2	Bob
00	10	None	Bob
01	10	e_1	Bob
10	10	e_2	Bob
11	10	e_2	Bob
00	11	None	Bob
01	11	e_1	Bob
10	11	e_1,e_2	Alice
11	11	e_1,e_2	Alice

Figure 3.4: Table used for checking the correctness of the protocol in Figure 3.3

As for the protocol computing 2-bit equality in 3.3, note that whenever Bob sees that no pipes became wet, he learns $x = 00$, and if Bob sees that only e_1 became wet, he learns $x = 01$. In both cases Bob is able to compute $EQ(x, y)$. If Bob sees that only e_2 became wet, he knows $x = 10 \vee x = 11$, but Bob can not distinguish between the two cases, so Bob can not check whether x equals y if $y = 10$. The function value (of equality) is not the same in the cases $(x = 10, y = 10)$ and $(x = 11, y = 10)$, but in both cases only e_2 became wet and the input of Bob is $y = 10$. Hence this protocol is not correct.

A simple protocol that correctly computes EQ_2 in the Leaky Garden-hose model is the following.

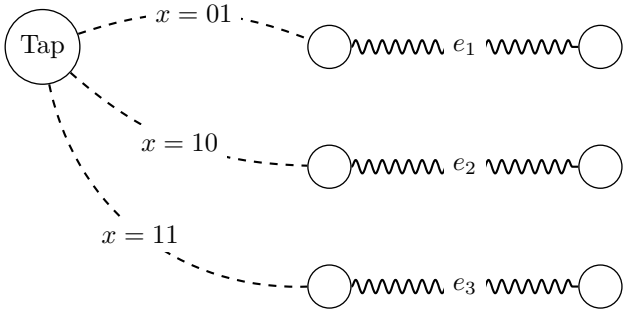

Figure 3.5: Correct Leaky Garden-hose protocol for 2-bit equality.

In the protocol of Figure 3.5, the pattern of wet pipes tells Bob exactly the value of Alice's input x.

x	y	Wet pipes	Who learns $EQ(x, y)$
00	00	None	Bob
01	00	e_1	Bob
10	00	e_2	Bob
11	00	e_3	Bob
00	01	None	Bob
01	01	e_1	Bob
10	01	e_2	Bob
11	01	e_3	Bob
00	10	None	Bob
01	10	e_1	Bob
10	10	e_2	Bob
11	10	e_3	Bob
00	11	None	Bob
01	11	e_1	Bob
10	11	e_2	Bob
11	11	e_3	Bob

Figure 3.6: Table used for checking the correctness of the protocol in Figure 3.5

An open problem at this point is whether there exists a Leaky Garden-hose protocol for 2-bit equality that uses two pipes, or whether there exists a Leaky Garden-hose protocol for 3-bit equality that uses only three or four pipes.

3.1.3 Intermediate models: bridging the gap

Since the Leaky Garden-hose model is more efficient than the original Garden-hose model, the question arises whether there exist weaker versions of the Leaky Garden-hose model that have a complexity between $LGH(f)$ and $GH(f)$. Such weaker models could prove valuable in showing interesting relationships between the Garden-hose model and the Leaky Garden-hose model, allowing a better understanding of the underlying mathematics.

For example, one could consider a model where the auxiliary output is the combination of where the water spills and a small part of the bit string $LGH_P(x, y)$, such that Alice and Bob learn which pipes became wet, but only for a limited number of pipes. The players thus receive less information about the flow of the water after the water has spilled. We could define the *s-Limited Leaky Garden-hose model* (for brevity of notation, *the $LLGH_s$ model*) in the following way. Define a parameter $s \in \mathbb{N} \cup 0$ that denotes the number of pipes that Alice or Bob are allowed to measure to have become wet or dry on input (x, y). Alice and Bob can decide, each on their own, which pipes they will measure, by requesting an s-bit substring of $LGH_P(x, y)$. For values of s bigger than the length of $LGH_P(x, y)$, Alice and Bob simply request the entire string $LGH_P(x, y)$. Note that this means that for $s \geq LGH(f)$ it holds that $LLGH_s(f) = LGH(f)$, as the s-Limited Leaky Garden-hose model then reduces to the Leaky Garden-hose model by construction.

Analogously to the Leaky Garden-hose model, we say *a protocol P computes a function f in the $LLGH_s$ model* if, after the water spills somewhere on input (x, y), and Alice and Bob each have learned an s-bit substring of $LGH_P(x, y)$, either Alice or Bob is able to compute with certainty the value of the function $f(x, y)$. Their computation of $f(x, y)$ can use the s-bit substring $LGH_P(x, y)$ that they requested, the information of where the water has spilled (Alice's or Bob's side) and their own input. We prove that the $LLGH_s$ model is indeed an intermediate model.

Lemma 25. *Let $LLGH_s(f)$ be the minimum number of pipes needed for a protocol to compute f in the $LLGH_s$ model. Then it holds for all $s \in \mathbb{N}$ and all Boolean functions f that*

$$LGH(f) \leq LLGH_s(f) \leq GH(f)$$

Proof. Note that in the $LLGH_s$ model, Alice and Bob are allowed to learn at whose side the water spills. This means that any Garden-hose protocol can be transformed to an s-Limited Leaky Garden-hose protocol using the same number of pipes, thus showing $LLGH_s(f) \leq GH(f)$.

Then, in order to conclude $LGH(f) \leq LLGH_s(f)$, we prove that for all Boolean functions f and all $s' \leq s'' \in \mathbb{N}$ it holds that $LLGH_{s''}(f) \leq LLGH_{s'}(f)$. This is done by observing that each protocol in the $LLGH_{s'}$ model can be transformed to a protocol in the $LLGH_{s''}$ model: Alice and Bob simply request the same s'-bit substrings of $LGH_P(x, y)$, and disregard the extra bits that they are allowed to request, because of s'' being greater than s'. □

A model similar to the s-Limited Leaky Garden-hose model has not been studied, and could provide new tools to create better bounds on the Garden-hose complexity.

3.2 The Time Bounded Garden-hose Model

Introduced in Klauck and Podder [2014], the following Time Bounded Garden-hose model is an adaptation of the Garden-hose model in which the time it takes before the water spills is limited. The notion of time in a Garden-hose model is defined in Klauck and Podder [2014] in the following way.

Definition 8. Let P be a Garden-hose protocol that computes a function $f(x, y)$. Define T_P to be the maximum number of pipes that get wet in P, maximizing over all inputs x, y.

Assuming the water flows equally fast in all pipes and all hoses, T_P intuitively describes the maximum time it takes for the water to flow through all the pipes and to spill at either Alice's side or Bob's side.

This leads to the question of classifying functions not by the minimum number of pipes needed to compute a function, but according to how fast the 'fastest' protocol computes a function. That is, instead of looking at the Garden-hose

complexity of a function f, we can examine the Garden-hose time-complexity of a function f by minimizing T_P over all protocols P that compute f. However, each function f has a protocol that computes it with $T_P \leq 2$, as exhibited by the protocol of Lemma 7 in Section 1.3. A different perspective yields more interesting results: considering only those protocols P with a T_P bounded by some number k, what is the complexity of a function f?

Definition 9. Let $GH_k(f)$ be the complexity of an optimal Garden-hose protocol P for computing f, while restricting P such that for every input (x, y) it must hold that $T_P \leq k \in \mathbb{N}$.

It turns out a time-size hierarchy follows, as is shown in Theorem 27 in Klauck and Podder [2014]. This roughly means that for some functions f, it holds that decreasing the allowed time k dramatically increases the number of pipes necessary to compute f. Details can be found in Klauck and Podder [2014].

3.2.1 Connection with the s-Limited Leaky Garden-hose model

Observe that the time complexity T_P of a protocol is exactly equal to the Hamming weight of $LGH_P(x, y)$, when maximizing over all input combinations x, y. The maximum number of pipes that get wet equals the length of the longest water path.

Remark. Let $wt(x)$ denote the Hamming weight of a bit string x. For all protocols $P = (m, f_A, f_B)$, it holds that

$$\max_{x,y \in \{0,1\}^n} wt(LGH_P(x, y)) = T_P$$

Although the relation between the s-Limited Leaky Garden-hose model and the Time Bounded Garden-hose model has not yet been studied in detail, our intuition leads us to believe there are strong connections to be found between $LLGH_s(f)$ and $GH_k(f)$ for $s = k$.

Chapter 4

Open Problems

In our study of extensions of the Garden-hose model we have mostly focused on the complexity of the equality function, as was also done in Chiu et al. [2014] for the Garden-hose model. The reasoning behind this is that even though the equality function is very simple in its definition, it elicits complex protocols. And while the various results in this thesis have allowed us to compare the new models with the Garden-hose model, this focus on equality has left behind an opportunity for further investigation on the complexity of different functions in the new models. For example, in Klauck and Podder [2014], the authors show various results relating Garden-hose complexity and function composition. An open question is whether those proofs can be translated to proofs about the other models in this thesis. We list several other remaining open questions that have arisen in this thesis below.

- What is the complexity of the n-bit, k-player equality function in the Multiparty Garden-hose model? And similarly for the Leaky Garden-hose model, what is the value of $LGH(EQ_n)$ for $n \geq 3$?
- Do there exist functions of exponential complexity in the Multiparty Garden-hose model?
- Does the proof of Lemma 12 in Section 1.3 translate to a proof of the statement $MPGH(f) \geq N_k(f) - 1$? Here, N_k is the multiparty (number-in-hand) nondeterministic communication complexity of f.
- In what ways can one use the proof technique suggested in Section 2.4.3? Can one use this technique from the Multiparty Garden-hose model to find new results in other models?
- What upper bounds hold for the complexity of functions in the s-limited Leaky Garden-hose model?
- What connections exist between the s-Limited Leaky Garden-hose model and to the Time Bounded Garden-hose model?

- What other results about the Garden-hose model can we translate to results in the new models?

Chapter 5

Bibliography

Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

Salman Beigi and Robert König. Simplified instantaneous non-local quantum computation with applications to position-based cryptography. *New Journal of Physics*, 13(9):093036, 2011.

Harry Buhrman, Nishanth Chandran, Serge Fehr, Ran Gelles, Vipul Goyal, Rafail Ostrovsky, and Christian Schaffner. *Position-Based Quantum Cryptography: Impossibility and Constructions*, pages 429–446. Springer Berlin Heidelberg, 2011.

Harry Buhrman, Serge Fehr, Christian Schaffner, and Florian Speelman. The garden-hose model. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 145–158. ACM, 2013.

Nishanth Chandran, Vipul Goyal, Ryan Moriarty, and Rafail Ostrovsky. *Position Based Cryptography*, pages 391–407. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Well Y. Chiu, Mario Szegedy, Chengu Wang, and Yixin Xu. *The Garden Hose Complexity for the Equality Function*, pages 112–123. Springer International Publishing, 2014.

S. Chowla, I. N. Herstein, and W. K. Moore. On recursions connected with symmetric groups. 3:328–334, 1951.

Hartmut Klauck and Supartha Podder. New Bounds for the Garden-Hose Model. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 481–492. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014.

Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

Neil J. A. Sloane and Simon Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.

Florian Speelman. Position-Based Quantum Cryptography and the Garden-Hose Game. Master's thesis, University of Amsterdam, 2011.