Epistemic Logics for Cryptographic Protocols and Zero-Knowledge Proofs

MSc Thesis *(Afstudeerscriptie)*

written by

Mateo C. Jaramillo
(born June 23rd, 1997 in Pasadena, California)

under the supervision of **Dr Aybüke Özgün** and **Prof Dr Hans van Ditmarsch**, 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 15, 2021	Dr Aybüke Özgün
	Prof Dr Hans van Ditmarsch
	Dr Malvin Gattinger
	Dr Christian Schaffner

Institute for Logic, Language and Computation

Abstract

This thesis presents an epistemic logic for modeling Zero-Knowledge proofs and other cryptographic protocols. We consider multi-agent interactions where a prover convinces a verifier of some proposition φ, with the verifier learning nothing more than the validity of φ. By enriching existing cryptographic variants of DEL with notions of probability, we introduce a cryptographic probability logic (CPL) that is capable of modeling cryptographic primitives such as encryption and passive adversaries while also capturing the concepts of interactive proofs and simulations. Within our presentation, we analyze multiple protocols including RSA encryption and a zero-knowledge proof for the NP-Complete problem of graph 3-coloring.

Acknowledgements

Firstly, I would like to thank my supervisors: Aybüke Özgün and Hans van Ditmarsch. They provided invaluable feedback on both the small technical details and the overall conceptual goal of the thesis. I would also like to extend my gratitude to the members of my defense committee Christian Schaffner and Malvin Gattinger for being experts in the field of cryptography and model checking and providing the level of critical scrutiny that challenged my thesis.

The intersection between cryptographic protocols and dynamic epistemic logic has been the topic of my interests during my time at the ILLC. I would like to thank Floris Roelofsen for entertaining my first attempt at tackling this problem during his Logic and Conversation class. That initial project functioned as a proof of concept for the work presented in this thesis. I would also like to the express my appreciation to Yfke Dulek for supervising a second attempt at researching this intersection. Her expertise in cryptography challenged just how robust the symbolic models would need to be, as well as introducing me to several concepts in cryptography that are fundamental to proving security.

I would like to thank my family and friends for their support during the entire Master of Logic program and thesis writing process. I would also like to thank Dr. Bruno Jacinto for introducing me to the ILLC and recommending that I continue studying logic beyond my initial undergraduate exposure. Finally, I want to thank the city of İstanbul for being home during my final year of study and providing a physical escape from the sometimes overwhelming writing process.

Contents

Contents **4**

1 Overview and Introduction **5**

2 Preliminaries **9**
2.1 Epistemic Logic 9
2.2 Dynamic Epistemic Logic 11
2.3 Cryptography 13
2.4 Probability Theory 15

3 A Logic for Cryptographic Protocols **17**
3.1 Syntax and Semantics 17
3.2 Cryptographic Protocols 21
3.3 Review and Discussion 36

4 A Logic for Zero-Knowledge Protocols **38**
4.1 Introduction 38
4.2 Zero-Knowledge Proofs 39
4.3 Cryptographic Probability Logic 43
4.4 Zero-Knowledge Protocols 55

5 Conclusion **69**

Bibliography **71**

CHAPTER 1

Overview and Introduction

In this thesis, we aim to examine cryptographic protocols through epistemic logics. We utilize a logic introduced in Dechesne and Wang [2007], specifically for reasoning about cryptographic protocols, and enhance features, such as the encryption keys, in order to model signature schemes. We extend the framework from Dechesne and Wang [2007] and from Chen and Deng [2020] with probabilities (inspired by a probability logic introduced in van Eijck and Schwarzentruber [2014]). With this enriched cryptographic probability logic, we are able to model zero-knowledge protocols.

A cryptographic protocol is a sequence of actions that ensure some security properties of a communication protocol are satisfied. These security properties can include secrecy, anonymity, privacy, authentication, or any combination of these properties. Protocols generally exist in multi-agent systems, so multiple agents may exchange information with some assurance that these aforementioned properties hold.

Existing work [Dechesne and Wang, 2007, 2010, Wang, 2010, Chen and Deng, 2020] has been productive in verifying security properties of certain cryptographic protocols using models in Dynamic Epistemic Logic (DEL). Given DEL's dynamic, multi-agent capabilities, it is capable of expressing intricate communication patterns as well as tracking the epistemic states of certain agents, which makes it a useful tool for verifying what information states hold after certain sequences of actions. The actual verification can be expressed by means of formal sentences, whose validity can be checked within the given model where the actions were executed. For example, after a cryptographic protocol in which Alice passes a secret message to Bob, we can verify that the intruder Eve (eavesdropping) does not know the contents of the secret message.

The logic of Dechesne and Wang and further developed by Chen and Deng allows us to model encryption of messages and adversarial actions. We provide several demonstrations of how we can model protocols in the logic as well as how we can design action models to represent adversaries of differing strengths. We also extend the logic by introducing the tools for asymmetric encryption keys, allowing us to model RSA signature schemes in the logic. In a subsequent extension, we add lotteries to the epistemic models, allowing agents within the model to assign probabilities to propositions. This cryptographic logic with probability is sufficient for modeling interactive proofs and zero-knowledge protocols.

A zero-knowledge protocol is an interaction between a prover and a verifier, in which the prover convinces the verifier of the validity of some statement without leaking any information beyond the validity of that statement [Goldwasser et al., 1989]. Therefore after a zero-knowledge protocol, the verifier is convinced of a statement but does not have the necessary information to prove the validity of the statement outside of the interaction with the prover. Zero-knowledge

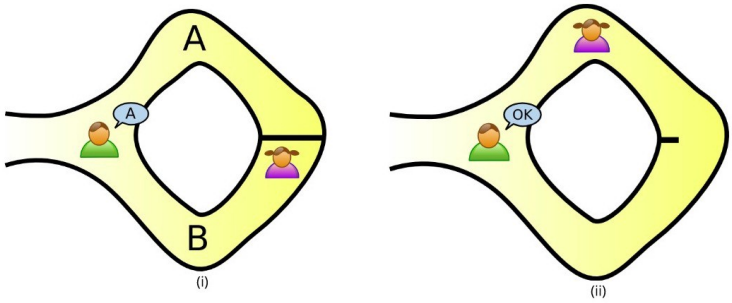


Figure 1.1: Visualisation of Ali Baba's cave [Commons, 2020]

interactions are epistemically interesting because of their counter-intuitive nature in which an agent knows that some sentence is valid, yet lacks the information proving its validity.

In later chapters of this thesis we use DEL variants to model the interaction between a prover and a verifier, as well as formalizing security goals within the syntax of the logic. Therefore, we can check the validity of the security goals for a protocol by designing formal sentences which accurately represent the security goals of the protocol and checking whether they are valid in a model after executing the actions of the protocol. More specifically, we can verify that the information agents obtained during the protocol does not violate zero-knowledge. Beyond verification, creating formal models of the protocols will allow an unobfuscated view to the logical structure of the protocol, separated from cryptographic primitives like numerical encryption procedures.

Zero-knowledge protocols can vary in design depending on the type of sentence they are attempting to validate. For instance in the Ali Baba cave example [Quisquater et al., 1990], a prover is trying to convince a verifier that she knows the password to a door obstructing a circular hallway where the two exits meet at a central point. So, the prover steps into the hallway, and the verifier randomly chooses an exit for the prover to walk to. If the prover has the password to the obstructing door, then this poses no challenge and the prover will always walk from the chosen exit, while if the prover does not have the password, then she can only correctly walk from the chosen exit with a 50% success rate. An illustration of this example and interaction can be seen in Fig 1.1. After a sufficiently large number of iterations, the verifier is convinced with high certainty that the prover does in fact know the password to the door, since in the alternative case, the probability of her being successful in the protocol approaches a negligible number. This protocol succeeds in convincing the verifier that the prover has the password without revealing any information about the password. Another example of a zero-knowledge protocol involves two friends Alice and Bob, where Bob is colorblind and Alice wishes to convince Bob that two dimensionally-identical balls are in fact distinguishable by their color. Since the balls are indistinguishable to Bob, it would not suffice for Alice to merely tell him that one is red and the other is green. Instead, they devise a zero-knowledge protocol in which Bob places both balls behind his back away from Alice. He then brings one ball forward to display, and then again places it behind his back. From here, Bob can choose to swap the balls between his hands or to do nothing. After the action (or lack of action) is executed, he brings forward his hand again and asks 'Did I switch the balls?'. If the balls are distinguishable then Alice will very easily answer 'Yes' or 'No' based on the decision Bob made, but if the balls are in fact indistinguishable and Alice is trying to deceive Bob, then she could still guess as to whether Bob made a switch. However, this guess would only be correct with a success rate of 50%, and therefore after a sufficiently large number of iterations, Bob is convinced of the distinguishability of the balls.

Clearly, these two protocols require different structures, parameters, and interaction types. Thus in order to capture a family of protocols which is extensive and useful, a quick aside to complexity classes must be made. The class of **NP** is used to define the class of decision problems for which a *witness* that shows the positive answer to the problem, can be verified in polynomial time. Formally we can consider a decision problem $\chi \in \mathbf{NP}$ and $x \in \chi$ iff there is a relation R_χ such that $\langle x, w \rangle \in R_\chi$ where w is the witness to the fact that $x \in \chi$ and the validity of w can be found in polynomial time [Arora and Barak, 2006]. A polynomial time reduction from a problem P to another problem χ means that if an algorithm exists for solving χ, then an algorithm exists for solving P as well. In other words, a reduction shows that one problem is only as difficult to solve as another problem. Furthermore, a problem χ is said to be **NP**-complete if every problem in **NP** is reducible in polynomial time to χ [Sipser, 2013].

For ease of explanation, later chapters of this thesis will focus on the **NP**-complete problem of Graph 3-colorability. A graph is said to be 3-colorable if it is possible to color the vertices of the graph such that no two vertices connected by an edge have the same color. Determining whether or not a graph is 3-colorable is **NP**-complete [Goldreich et al., 1987]. The example used in Chapter 4 of this thesis is a zero-knowledge protocol in which a prover convinces a verifier that a graph is 3-colorable without revealing the coloring of the graph which serves as the witness. It will be sufficient to show how to model this protocol as well as verify the security goals, since all other problems within **NP** are reducible to the Graph 3-colorability problem [Goldreich et al., 1991].

Consider a scenario between a government employee Alice trying to verify her identity with an unknown agent Bob whose allegiance is disputed. Let this government agency's identification be a graph and a satisfying 3-color assignment to its vertices. Alice does not want to reveal the coloring she knows to her graph, because if he is a foe then he can take her graph and coloring to impersonate her, and regardless if Bob is friendly, Alice is not sure whether there any eavesdroppers on the communication channel. So, together they implement a zero-knowledge protocol in which Bob will be convinced of the validity that her graph is 3-colorable without learning what that coloring is, nor will Bob be able to learn any coloring for the graph. The zero-knowledge protocol will be an interaction between the two agents: Alice who is the prover and Bob who is the verifier. Alice's goal will be to convince Bob that a particular graph is 3-colorable, which she knows because she knows a coloring for the graph that satisfies the definition of 3-colorability. Bob's goal is two-fold: if the graph is indeed 3-colorable, then Bob wants to learn the validity of this statement; if the graph is not 3-colorable then the probability that Bob is convinced otherwise is negligible. Bob's first goal is commonly called *Completeness* and his second goal is called *Soundness*. Completeness and Soundness are common properties of interactive protocols. By the end of the protocol, Bob is convinced that the graph is 3-colorable, yet he will not know the coloring that Alice knows which justifies her knowledge of this statement. Furthermore, Bob will not possess any evidence which might enable him to deduce the coloring Alice knows. This last property is referred to as *Zero-Knowledge*.

Overview of Previous Works The initial cryptographic variant of DEL in Chapter 3 was originally introduced in Dechesne and Wang [2007] and later improved in Chen and Deng [2020]. The protocol examined in Chen and Deng [2020] used symmetric key encryption and a passive adversary. We examine the protocol used in Chen and Deng [2020] and we analyze the protocol under a stronger adversary as well as implementing asymmetric keys in the logic. The asymmetric keys allow us to model signature schemes within the logic and to verify security properties about the scheme.

The major novel contribution of this thesis lies in Chapter 4 with the introduction of a cryptographic probability logic that can model zero-knowledge protocols. Although there have

been many symbolic systems presented with the capabilities of modeling zero-knowledge protocols, they rely on complicated term algebras and complex primitives [Backes et al., 2015, 2007, Baskar et al., 2009]. The concept of using epistemic logics to model zero-knowledge interactions has been previously investigated by Halpern et al. [1988] and Halpern et al. [2009], however these works characterize the requirements of zero-knowledge in an epistemic temporal logic without focusing on protocol specifications. Our logic presented in Chapter 4 provides a characterization of the zero-knowledge requirements as well as providing a general schematic to modeling a protocol and verifying whether these requirements hold. This is in large part possible due to a creative use of epistemic action models to represent simulated protocols with rewind capabilities.

Outline The thesis will be structured in the following manner: first, preliminary notions regarding epistemic logics, cryptography, and probability shall be introduced. Next, we will examine an existing cryptographic variant of dynamic epistemic logic capable of modeling cryptographic protocols and verifying security features of the protocol. We will demonstrate how this logic represents the actions of adversaries in a modular way that allows us to modify the strength of the adversary without changes to the formal modeling mechanisms or the underlying protocol structure. Finally, we will enrich this cryptographic variant with lotteries which will allow the agents in the model to use probabilistic reasoning. This logic will be sufficient for formalizing the conditions of a zero-knowledge protocol, and thus will allow us to apply the enriched logic to a zero-knowledge protocol for graph 3-colorability.

CHAPTER 2

Preliminaries

The purpose of this chapter is to introduce the preliminary concepts which we will be developing upon within this thesis. These concepts include dynamic epistemic logic, cryptographic protocols, and basic probability theory. The later chapters include relevant definitions with modifications as needed. We do presuppose familiarity with propositional logic, modal logic, and standard set-theoretic notation.

2.1 Epistemic Logic

Epistemic logics allow us to reason about multi-agent systems in modal logic and express statements about the epistemic states of a given agent. Given a set of agents $A = \{i_1, ..., i_n\}$, then we say i knows a proposition φ when φ is true in all worlds that are accessible by i. In that way, knowledge works similarly to the $\Box$ operator of propositional modal logic. We first define the epistemic models of the logic before defining the syntax of epistemic logic. We then proceed to introduce the semantics of the language which will allow us to reason about statements in the language.

2.1.1 Kripke Frames

Modal logic uses the concept of worlds in order to make statements of necessity and possibility. Worlds can be connected within a Kripke frame which is denoted as a tuple (W, R). For our purposes, it is sufficient to only consider finite models. Let W be a finite set of worlds which describe different states and R be an accessibility relation over the set of worlds. Two worlds w and v are accessible by one another if $\langle w, v\rangle \in R$, we can also write wRv for simplicity. To use a Kripke frame for reasoning, we must pair it with a valuation function which assigns to each world a set of propositions, namely those propositions which are true at that world.

Definition 1 (Kripke Model)**.** *Let (W, R) be a Kripke frame, $\mathcal{V}$ be a valuation function, and $\mathbf{P}$ be a finite set of propositions. We define a Kripke model $\mathcal{M}$ as the tuple $(W, R, \mathcal{V})$ such that $\mathcal{V} : W \to \wp(\mathbf{P})$.*

By defining the accessibility relation R as a function which takes as input the set of agents A and outputs pairs of worlds, we can define a family of accessibility relations for each agent in A over the set of worlds in W.

Definition 2 (Multi-Agent Kripke Model)**.** *Let A be a finite set of agents, W be a finite set of worlds, and $\mathcal{V} : W \to \wp(\mathbf{P})$ be a valuation function. $\mathcal{M} = (W, R, \mathcal{V})$ defines a multi-agent Kripke model where $R : A \to \wp(W \times W)$ is a function which assigns to each agent $i \in A$ a set of pairs of worlds. Let wR_iv denote the case that $\langle w, v\rangle \in R(i)$ or informally, that the world v is reachable from the world w by the agent i. Furthermore, a pointed-model is a model $\mathcal{M}$ with a world $w \in W$ designated as the* actual *world. Pointed models are denoted as the pair $\mathcal{M}, w$ where w is the actual world of $\mathcal{M}$.*

We can define features about a Kripke model given certain properties of the accessibility relations. R is reflexive if every world is accessible from itself. R is symmetric if for all pairs wR_iv, it also holds that vR_iw. Lastly, R is transitive if wR_iv and vR_iz, then wR_iz. Thus, if R is reflexive, symmetric, and transitive, then R is an equivalence relation over the worlds for each agent.

2.1.2 Syntax and Semantics

For the models, we need a language powerful enough to express epistemic statements with respect to agents in the model. Let $\mathcal{L}$ be the language of epistemic logic with the knowledge operator K.

Definition 3 (Syntax of $\mathcal{L}$)**.** *Let p range over a finite set of propositions $\mathbf{P}$ and i range over the set of agents A. We define the language $\mathcal{L}$ as follows in BNF:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid K_i\varphi$$

This BNF describes the ways in which formulas of φ can be defined. $\top$ is the constant for an always true statement or tautology. $\neg\varphi$ denotes the negation of a formula φ. $\varphi \wedge \varphi'$ is the conjunction of two formulas φ and φ'. The last formula denotes: agent i knows that φ.

From the above definition we can define more convenient connectives with common abbreviations.

$$\begin{array}{rclcrcl} \bot & := & \neg\top & & \varphi \vee \varphi' & := & \neg(\neg\varphi \wedge \neg\varphi') \\ \varphi \to \varphi' & := & \neg\varphi \vee \varphi' & & \varphi \leftrightarrow \varphi' & := & (\varphi \to \varphi') \wedge (\varphi' \to \varphi) \end{array}$$

The above abbreviations define a contradiction which is always false, disjunction, material implication, and equivalence, respectively.

With the syntax established and the models defined we can give the semantics for $\mathcal{L}$.

Definition 4 (Semantics)**.** *Let $\mathcal{M} = (W, R, \mathcal{V})$ be a multi-agent Kripke model and $w \in W$.*

$$\begin{array}{ll} \mathcal{M}, w \vDash \top & \Leftrightarrow \ \textit{always} \\ \mathcal{M}, w \vDash p & \Leftrightarrow \ p \in \mathcal{V}(w) \\ \mathcal{M}, w \vDash \neg\varphi & \Leftrightarrow \ \mathcal{M}, w \nvDash \varphi \\ \mathcal{M}, w \vDash \varphi \wedge \varphi' & \Leftrightarrow \ \mathcal{M}, w \vDash \varphi \textit{ and } \mathcal{M}, w \vDash \varphi' \\ \mathcal{M}, w \vDash K_i\varphi & \Leftrightarrow \ \textit{if } wR_iv, \textit{ then } \mathcal{M}, v \vDash \varphi \end{array}$$

φ is true at w in $\mathcal{M}$ iff $\mathcal{M}, w \vDash \varphi$ and φ is true in $\mathcal{M}$, denoted as $\mathcal{M} \vDash \varphi$, iff φ is true at all $w \in W$ in $\mathcal{M}$.

By the above definition, an agent i knows a proposition φ if φ is true at all worlds accessible by i.

2.2 Dynamic Epistemic Logic

The models described so far represent static models, or models in which we are presented with a snapshot of the current epistemic situation. To reason about puzzles, communication, and protocols we need to be able to represent the situations in which information changes in a model. The forms of information change might involve all agents learning the truth value of some proposition, which would update their accessibility relations, or the agents execute some action within the model which changes the structure of the model itself. These actions might be a form of private communication, or in more advanced action models it could be the reassignment of a truth value for a proposition. An epistemic logic with semantics designed for dynamic events is called a dynamic epistemic logic.

As we have already described static Kripke models, we now need to define a general action model, which when applied to a Kripke model will result in an updated model. These updates will represent the dynamic change of information.

An action model structure is similar to a Kripke model structure, with the exception that worlds are now action labels and the valuation function is replaced with preconditions and postconditions. The postconditions will be defined by substitution functions adopted from van Ditmarsch and Kooi [2008]. If $\mathbf{P}$ is the set of basic propositions for the language $\mathcal{L}$, then the function $\mathbf{P} \to \mathcal{L}$ is a substitution in $\mathcal{L}$ if it maps all but a finite number of basic propositions to themselves. The set of all substitutions for $\mathbf{P}$ in $\mathcal{L}$ is denoted by $\mathsf{SUB}(\mathcal{L})$.

Definition 5 (Action Model)**.** *An action model $\mathfrak{A}$ is a tuple $(\mathcal{A}, \sim, \mathsf{Pre}, \mathsf{Pos})$ where $\mathcal{A}$ is a finite set of action labels $\mathcal{A} = \{\sigma_1, ..., \sigma_n\}$, $\sim$ is a family of accessibility relations $\sim_i$ over the actions $\sigma \in \mathcal{A}$ for $i \in A$ such that $\sigma \sim_i \sigma'$ iff agent i cannot distinguish between the execution of σ and σ'. The precondition function $\mathsf{Pre} : \mathcal{A} \to \mathcal{L}$ assigns to each action σ a formula φ such that φ is true in a world iff σ can be executed at that world. The postcondition is a function $\mathsf{Pos} : \mathcal{A} \to \mathsf{SUB}(\mathcal{L})$ which assigns to each action $\sigma \in \mathcal{A}$ a substitution from the basic propositions to formulas in the language $\mathcal{L}$. This allows the postcondition of an action to change the truth valuation of a proposition.*

We can now make an addition to the syntax of Definition 3 by allowing formulas to have a prefix denoting the execution of an action in an action model $\mathfrak{A}$. We refer to the language $\mathcal{L}$ that is capable of expressing statements about dynamic actions as $\mathcal{L}_{DEL}$, for dynamic epistemic logic.

Definition 6 (Syntax of $\mathcal{L}_{DEL}$)**.** *Let p range over the propositions of $\mathbf{P}$, i over the agents in A, and σ over the actions in $\mathfrak{A}$. The formulas φ of the language $\mathcal{L}_{DEL}$ are defined as follows:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid K_i\varphi \mid [\mathfrak{A}, \sigma]\varphi$$

The dynamic formula $[\mathfrak{A}, \sigma]\varphi$ denotes that after the action σ in $\mathfrak{A}$ is executed, φ holds.

We define the semantics of $\mathcal{L}_{DEL}$ the same as the semantics of static epistemic logic with an additional clause for the new dynamic operator.

Definition 7 (Semantics of $\mathcal{L}_{DEL}$)**.**

$$\mathcal{M}, w \vDash [\mathfrak{A}, \sigma]\varphi \quad \Leftrightarrow \quad \text{if } \mathcal{M}, w \vDash \mathsf{Pre}(\sigma) \text{ then } \mathcal{M} \circ \mathfrak{A}, \langle w, \sigma\rangle \vDash \varphi$$

where $\mathcal{M} \circ \mathfrak{A}$ is the update model composed of $\mathcal{M}$ and $\mathfrak{A}$.

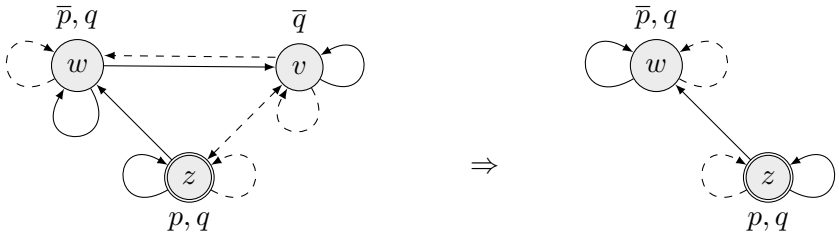


Figure 2.1: Public announcement of q in a DEL model $\mathcal{M}$

An update model is the model that results from updating a Kripke model with an action model. An update model also has a similar structure to the structure of static Kripke models with worlds being replaced by world and action pairs, denoting the actions which can be executed at a given world. Furthermore, the accessibility relations of both the initial model and the action model are combined such that world and action pairs are indistinguishable from other world and action pairs iff both the worlds were indistinguishable in the initial model and both the actions were indistinguishable in the action model. Lastly, the valuation function uses the reassignments of the postcondition of an action in order to determine the new valuation for a given world in the update model.

Definition 8 (Update Model)**.** *Let $\mathcal{M} = (W, R, \mathcal{V})$ be a multi-agent Kripke model and $\mathfrak{A} = (\mathcal{A}, \sim, \mathsf{Pre}, \mathsf{Pos})$ be an action model. The update model composed of $\mathcal{M}$ updated by $\mathfrak{A}$ is denoted as $\mathcal{M} \circ \mathfrak{A}$ which is defined as the tuple $(W', R', \mathcal{V}')$.*

- $W' := \{\langle w, \sigma\rangle \mid \mathcal{M}, w \vDash \mathsf{Pre}(\sigma)\}$
- $R' := \{R'_i \mid i \in A\}$ *such that* $(\langle w, \sigma\rangle, \langle v, \sigma'\rangle) \in R'_i$ *iff* $\langle w, v\rangle \in R_i$ *and* $\sigma \sim_i \sigma'$.
- $\mathcal{V}'(\langle w, \sigma\rangle) := \{p \mid \mathcal{M}, w \vDash \mathsf{Pos}(\sigma)(p)\}$

We can demonstrate an example of a model update with the following Kripke model and action model. Let $\mathcal{M} = (W, R, \mathcal{V})$ be a Kripke model with a set of agents $A = \{a, b\}$. $W = \{w, v, z\}$ is the set of worlds in the model with z being the actual world, and the accessibility relations and valuations for each world are illustrated in the leftmost model of Figure 2.1, where the solid arrows denote the accessibility relations of a and the dashed arrows denote those of b. Since both agents of the model cannot distinguish between the actual world z and another world in which the valuation of the propositions p and q are different from z, neither agent is said to *know* both propositions. However, $\mathcal{M}, z \vDash K_a q$ holds and $\mathcal{M}, z \vDash K_b p$ holds.

An example of one of the more basic actions is that of public announcement. A public announcement simply announces to all agents within the model that a given proposition holds. This action severs the accessibility relations of the agents to the worlds in the model in which the proposition does not hold. We can let $\mathfrak{A}$ be an action model with one action σ which will represent the announcement of q. The precondition of the proposition q is that q is true in the world in which it is announced. Since the public announcement of q makes no changes to the valuation of the propositions at a given world, we can denote the postcondition as id which signifies that $\mathcal{V}(w) = \mathcal{V}'(\langle w, \sigma\rangle)$ for all w such that $\mathcal{M}, w \vDash \mathsf{Pre}(\sigma)$.

By updating the model $\mathcal{M}$ with $\mathfrak{A}$, then the worlds of the updated model become the set of worlds such that $\mathsf{Pre}(\sigma)$ is true, so in the example of Figure 2.1, that is worlds w and z. The accessibility relations connecting the worlds of w and z to the world v where $\neg q$ held, are severed in the rightmost model of Figure 2.1.

Notably, even after the update, a still cannot distinguish between the worlds w and z, thus a does not know whether p is true. Since b's only reachable world from the actual world z is z (the reflexive relation), then b does know p and q, thus $\mathcal{M} \circ \mathfrak{A}, \langle z, \sigma \rangle \vDash K_b p \wedge K_b q$ holds.

2.3 Cryptography

Cryptography is the study of secure communication techniques and protocols. Many of the methods and protocols used in this thesis will abstract away from cryptographic primitives such as encryption functions and random bits, but as a useful guide as to what is being represented by operators in the later chapters, we explain some basic encryption procedures here.

Oftentimes the goal of cryptography is to create ciphers which are easy to decipher if the key is known and hard or infeasible if the key is not known. Much of modern cryptography relies on notions of computational infeasibility, which dictates how difficult a computation is for any computational device. For example, problems in the complexity class of **NP** are difficult for a computer to solve in polynomial time, but by definition a solution to a problem is verifiable in polynomial time. The security of many popular encryption methods will rely on this asymmetric difficulty. So, any potential adversary cannot use a polynomial time algorithm to attempt to invert the computations of the encryption without some auxiliary information. However, an authorized participant in the protocol, who has said auxiliary information, can compute the message in a reasonable amount of time. This often restricts potential attacks by adversaries to *brute force* attacks, which involves them guessing and checking solutions to a problem in the class of **NP**. For problems with large enough numbers and entities, the probability of a successful guess can diminish to near-impossibility.

2.3.1 Encryption Method Examples

Many of the encryption methods which we will investigate rely on the idea of one-way functions. As described above, one-way functions are easy to compute for any given input, but given the image of a random input, they are infeasible to invert. This section will briefly examine the mathematics behind the popular RSA encryption method as well as the underlying principles of a hash function.

RSA Encryption RSA is known as a public-key encryption system since it uses public information (the public key) known to all users and private information (the private keys) only known to individual users. RSA works by providing each user with a pair of pairs, one being her encryption key and the other being her decryption key. A user's encryption and decryption keys are pairs of positive integers (e, n) and (d, n), respectively. The integers e, d, n are selected in a way such that it is mathematically possible to encrypt a message M (which is a number) by applying a function of the encryption key on the message. First, n is a public number known to all users and n is computed as the product of two secret large primes p and q. Despite n being public, its considered a hard problem to factor n and obtain p, q [Katz and Lindell, 2007]. d is selected as a number that is relatively prime to $(p-1) \cdot (q-1)$ which means that the GCD[1]$(d, (p-1) \cdot (q-1)) = 1$ and e is the multiplicative inverse of $d \pmod{(p-1) \cdot (q-1)}$ so $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$. Encrypting a message M is $M^e \pmod{n}$ and similarly, decrypting a message M is $M^d \pmod{n}$. Since e and d are multiplicative inverses the process of encrypting and then decrypting a message is the same as decrypting and then encrypting the message. This can be seen in $(M^e)^d \pmod{n} = M^{e \cdot d} \pmod{n} = (M^d)^e \pmod{n}$.

[1] $\mathsf{GCD}(X, Y)$ denotes the greatest common divisor of X and Y.

While RSA is a popular public-key encryption method, there are also symmetric key encryption methods which are not reliant on public and private information, but only on privately held information. So, encryption with a symmetric key can be thought of as applying an operation onto a message M, like multiplication by y, and the act of decryption is simply the inverse operation of division by y. Therefore, the operation can be thought of as the symmetric key, and thus as long as it is kept secret, then the secret values being encrypted cannot be read by unauthorized users, or users who were not made privy to the key.

Hash Functions Hash functions take strings of bits of arbitrary length, potentially infinite, and compress them into strings of a fixed length. One of the basic requirements for a hash function is that it should be difficult to find a collision. A hash function H has a collision if there is a pair of distinct elements x and x' such that $H(x) = H(x')$. In such a case, x and x' are said to collide in H. H is *collision resistant* if it is infeasible to find a collision in H in polynomial time. Since hash functions take strings of arbitrary length and compress them to strings of fixed length, there must exist some collision due to the pigeon-hole principle [Katz and Lindell, 2007]. So once again, the security of the hash function H relies on the difficulty of finding values that collide in H.

2.3.2 Dolev-Yao Models

To analyze the security of cryptographic protocols, we must make a choice about the assumptions and proof methods of the model in which the analysis will take place. The computational model is traditionally chosen since all messages are represented by bit strings over the language $\{0,1\}^*$, and the principals and adversaries are defined as algorithms which follow a given strategy based upon their inputs. Proving security in the computational model typically relies on a proof by reduction, which assumes that an attack by an adversary is possible on the protocol, and then shows how this attack implies an attack on one of the underlying cryptographic primitives of the protocol such as the encryption method or the hash function. Therefore, by proving that the underlying primitives are impervious to attacks in polynomial time, the protocols are proven to be secure against probabilistic polynomial time algorithms.

However, the computational models are intricate systems based reliant on advanced mathematical operations and functions, and designing the reductions used can prove to be a difficult task in itself. A Dolev-Yao model [Dolev and Yao, 1983] is a symbolic model which represents messages as elements of a term algebra and represents the cryptographic primitives of encryption as operations of the algebra. Since all possible actions are defined by the operations and available messages known to each principal of the protocol in a Dolev-Yao model, it provides a compact method for defining any possible adversarial behavior. Therefore, this method of modeling makes it possible to reason about the security of a given protocol with respect to an adversary's attack strategies. A general theorem could prove that an adversary's goals lie outside the range of possible executable actions, where an action is a transfer of information from one principal to another or a deduction of information from available information. We do not investigate Dolev-Yao models within this thesis, rather we use DEL to model protocols, but it is important to identify the basic fundamentals of a symbolic model that would allow us to produce the most accurate representations of a protocol.

A general Dolev-Yao model for public-key encryption and symmetric key encryption would use an algebra $\mathbb{A}$ where the messages of the model are elements of $\mathbb{A}$. Fragments of $\mathbb{A}$ can define a set of publicly accessible users $\mathcal{I}$, a set of privately generated nonces $\mathcal{N}$, as well as three types of encryption keys: public, private, and symmetric, denoted as $\mathcal{K}_{Pub}, \mathcal{K}_{Priv}$, and $\mathcal{K}_{Sym}$, respectively. The encryption function can be an operation on $\mathbb{A}$ that takes a public key and a

message and outputs a message defined as the function $\mathsf{encrypt} : \mathcal{K}_{Pub} \times \mathbb{A} \to \mathbb{A}$. Likewise, a concatenation or pairing function can be defined by $\mathsf{pair} : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$.

Assuming a bijection between public and private keys, we can define an inversion function $\mathsf{invert} : \mathcal{K}_{Pub} \to \mathcal{K}_{Priv}$. Let $\mathcal{K}^{-1} = \begin{cases} \mathsf{invert}(\mathcal{K}) & \text{if } \mathcal{K} \text{ is a public key} \\ \mathsf{invert}^{-1}(\mathcal{K}) & \text{if } \mathcal{K} \text{ is a private key} \end{cases}$. Intuitively, the inverse of a symmetric key is itself, so $\mathcal{K}^{-1}$ would be the identity function [Herzog, 2005].

From the operations and terms defined above, we can construct protocol descriptions based on nuanced message distributions and principal assignments in $\mathcal{I}$. A protocol can then be defined as a sequence of messages in $\mathbb{A}$, where each message can have the prefix of the sender in $\mathcal{I}$, and the suffix of the intended receiver in $\mathcal{I}$ along with any other operations applied to the messages.

2.4 Probability Theory

In later chapters of this thesis, we will analyze zero-knowledge protocols which require principals in a protocol to be convinced of the validity of a proposition to a certain degree. That is, a verifier in a protocol will assign probability values to certain statements that define whether or not the verifier knows the statement. This section outlines basic probability notions which will be used in later chapters, and can be skipped by those familiar to probability theory. For a more thorough introduction to probability theory, see Schulz and Schaffner [2018].

Probability values are assigned to elements of a sample space, which can be defined as a set.

Definition 9 (Sample Space)**.** *Let any set Ω that can be formed from open sets through the operations of countable union, countable intersection, and relative complement be a sample space. Denote the members of a sample space as $\omega \in \Omega$.*

A sample space can be thought of as a collection of events. We can define these events as subsets of the sample space Ω.

Definition 10 (Event and Event Space)**.** *Any subset $E \subseteq \Omega$ is an event. An event space associated with a sample space Ω is a set $\mathcal{E}$ such that:*

1. $\mathcal{E} \neq \emptyset$
2. *If $E \in \mathcal{E}$, then $E \subseteq \Omega$*
3. *If $E \in \mathcal{E}$, then $\Omega \backslash E \in \mathcal{E}$*
4. *If $E, E' \in \mathcal{E}$, then $E \cup E' \in \mathcal{E}$*

We now define a probability measure as a function from an event space to a real number. The following properties are also referred to as the Kolmogorov Axioms.

Definition 11 (Probability Measure)**.** *Define a probability measure $\mathbb{P} : \mathcal{E} \to \mathbb{R}$ on an event space $\mathcal{E}$ associated with the sample space Ω. $\mathbb{P}$ is a probability measure iff $\mathbb{P}$ satisfies the following properties:*

1. $\mathbb{P}(E) \geq 0$ *for all* $E \in \mathcal{E}$
2. $\mathbb{P} \bigcup_{i=1}^{\infty} E_i = \sum_{i=1}^{\infty} \mathbb{P}(E_i)$ *for disjoint events* $E_1, E_2, ...$

3. $\mathbb{P}(\Omega) = 1$.

We can now define the notion of uniform probability, which is the probability measure for equally likely events. Thus the value $\frac{1}{|\Omega|}$ defines the uniform probability assigned to each $\omega \in \Omega$. So, the elements in our sample space Ω are equally probable when $\mathbb{P}(\{\omega\}) = \frac{1}{|\Omega|}$ for all $\omega \in \Omega$.

The probability of two events occurring together is defined as the *joint* probability of the two events. So, for two events E_i and E_j associated with Ω, $E_i \cap E_j = \Omega \backslash ((\Omega \backslash E_i) \cup (\Omega \backslash E_j))$ and the joint probability of E_i and E_j is simply $\mathbb{P}(E_i \cap E_j)$. We can now define the notions of conditional probability and independence with respect to events. Informally, a conditional probability is the probability measure of an event conditional on another specified event occurring.

Definition 12 (Conditional Probability)**.** *Let E_i and E_j be an event and $\mathbb{P}(E_j) > 0$, then the probability of E_i conditional on E_j is defined as*

$$\mathbb{P}(E_i|E_j) = \frac{\mathbb{P}(E_i \cap E_j)}{\mathbb{P}(E_j)}$$

The probability of E_i conditional on E_j is defined as the probability of both E_i and E_j occurring divided by the probability of E_j.

If two events are not related, in the sense that the occurrence of one event does not affect the occurrence of the other, then the two events are said to be independent. Formally, if two events E_i and E_j are independent, then $\mathbb{P}(E_i \cap E_j) = \mathbb{P}(E_i) \times \mathbb{P}(E_j)$.

Remark 1. *If two events E_i and E_j are independent, then the conditional probability of E_i on E_j is:*

$$\mathbb{P}(E_i|E_j) = \frac{\mathbb{P}(E_i \cap E_j)}{\mathbb{P}(E_j)} = \frac{\mathbb{P}(E_i) \times \mathbb{P}(E_j)}{\mathbb{P}(E_j)} = \mathbb{P}(E_i)$$

CHAPTER 3

A Logic for Cryptographic Protocols

This chapter focuses on using a variant of DEL to model cryptographic primitives such as encryption and adversarial actions. We expand upon the logic introduced in Dechesne and Wang [2007] by introducing asymmetric encryption keys, making the logic more in line with traditional Dolev-Yao models. We examine several cryptographic protocols as well as methods for verifying security properties.

3.1 Syntax and Semantics

The logic used will be a variant of DEL in which standard DEL notation and syntax will be used, but the propositions will be designed to better represent our cryptographic needs. Our language is denoted by $\mathcal{L}$ where A is a finite set of agents we want to analyze within the protocol. $\mathcal{A}$ will be used to denote the set of actions in an action model $\mathfrak{A}$. Since cryptography relies on the exchange of messages in a secure manner, we will define the set M as the set of messages.

Definition 13 (Messages)**.** *Let M be a finite set of messages. A message $m \in M$ is defined as:*

$$m ::= n \mid k \mid \{m\}_k \mid (m, m')$$

where n is an encoding of a message in ASCII, k is a cryptographic key, $\{m\}_k$ denotes a message m encrypted by key k, and (m, m') is the pairing of messages m and m'.

Following the work of Dechesne and Wang [2007] and Chen and Deng [2020] propositions within the logic are refined by three predicates: has, const, and mk. The predicate has will denote that an agent *has* a particular message within their information set, const will denote that the agent can *construct* the message from the information within their information set following a specified set of construction rules defined in Definition 17, and mk will *mark* actions, simultaneously defined in Definition 19, in the protocol that have been executed.

Definition 14 (Basic Propositions)**.** *The set Φ^A is the set of basic propositions defined by:*

$$p ::= \mathsf{has}_i m \mid \mathsf{const}_i m \mid mk(\sigma)$$

where $i \in A$, $m \in M$, and $\sigma \in \mathcal{A}$. The proposition $\mathsf{has}_i m$ denotes that the agent i possesses the message m, which means that i received the message m or obtained it during initial distribution. The proposition $\mathsf{const}_i m$ means that i can construct m from the messages that i possesses via

construction rules. Lastly, the proposition $mk(\sigma)$ denotes that the action σ is marked. Let $\Phi_I, \Phi_{\overline{I}}$, and Φ_{AL} be subsets of Φ^A that contain the above propositions, respectively. That is Φ_I contains only propositions of the form $\mathsf{has}_i m$ for all $i \in A$, $\Phi_{\overline{I}}$ contains only propositions of the form $\mathsf{const}_i m$ for all $i \in A$, and likewise for Φ_{AL} and propositions $mk(\sigma)$ for all $\sigma \in \mathcal{A}$.

Definition 15 (Syntax)**.** *Given agents A, propositions Φ^A, and actions $\mathcal{A}$, formulas of $\mathcal{L}$ are defined by:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid K_i\varphi \mid [\mathfrak{A}, \sigma]\varphi \mid [\mathfrak{A}]^*\varphi$$

Standard DEL meanings are assigned to each connective and operator, so $K_i\varphi$ means that "agent i knows φ is true" and $[\mathfrak{A}, \sigma]\varphi$ states that "after an action σ, φ holds in the model updated with $\mathfrak{A}$". The formula $[\mathfrak{A}]\varphi$ is an abbreviation for $\bigwedge_{\sigma \in \mathfrak{A}} [\mathfrak{A}, \sigma]\varphi$ and $[\mathfrak{A}]^*\varphi$ signifies that after a finite number of iterations of updating a model with $\mathfrak{A}$, φ holds. Standard abbreviations for disjunction, material implication, and equivalence as shown in Definition 3 are assumed.

We also use the following abbreviations for formulas involving action models: $\langle \mathfrak{A}, \sigma\rangle\varphi := \neg[\mathfrak{A}, \sigma]\neg\varphi$, $\langle \mathfrak{A}\rangle\varphi := \neg[\mathfrak{A}]\neg\varphi$, and $\langle \mathfrak{A}\rangle^* := \neg[\mathfrak{A}]^*\neg\varphi$.

Definition 16 (Models of $\mathcal{L}$)**.** *A model is a tuple $\mathcal{M} = (W, R, I, AL)$ where W is a nonempty, finite set of worlds. $R : A \to \wp(W \times W)$ is a function from the agents in A to a set of pairs of worlds, denoting the reflexive, transitive, and symmetric accessibility relation of each agent over the worlds in W. $I : W \times A \to \wp(M)$ are the information states such that $I_{w,i}$ is the set of information that i possesses at world w either by an initial distribution or by receiving them through communication with other agents. $AL : W \to \wp(\mathcal{A})$ are the action labels, so $AL(w)$ is the set of actions marked at w.*

I and AL function the same as valuations for the propositions of $\mathcal{L}$. Thus, the proposition $\mathsf{has}_i m$ is satisfied in a world w iff $m \in I_{w,i}$. The proposition $\mathsf{const}_i m$ is satisfied in a world w iff there is a derivation from some set $\Gamma \subseteq I_{w,i}$ to m using the rules of $\mathbb{C}$ defined below in Definition 17.

Definition 17 (Cryptographic Construction Rules $\mathbb{C}$)**.** *$\mathbb{C}$ is a set of construction rules of the form $\Gamma \vdash m$ where Γ is called a set of conditions and m is a result, it must be noted that $\Gamma \subseteq M$ and $m \in M$. The set $\mathbb{C}$ acts as a set of rules that the agents can use to derive messages from their information sets. The following rules determine what messages an agent can construct from his information set. For ease of readability, the conditions of the rule are listed above the line and the result of the rule is listed below the line.*

$$\frac{m \quad k}{\{m\}_k} \qquad \frac{\{m\}_k \quad k}{m} \qquad \frac{m \quad m'}{(m, m')} \qquad \frac{(m, m')}{m} \qquad \frac{(m, m')}{m'} \qquad \frac{m}{m}$$

An agent can only use a rule if his information set contains its conditions. Intuitively, with a message and an encryption key one can construct an encrypted form of the message. Likewise, with an encrypted message and the corresponding key one can obtain the plaintext message. The subsequent three rules pertain to concatenation introduction and elimination and the final rule states that one can always construct what they possess. Given an information set $I_{w,i} \subseteq M$, $\overline{I_{w,i}}$ is the closure of $I_{w,i}$ under $\mathbb{C}$.

First we will define the semantics of the basic propositions has, const, and mk before we move on to the semantics of the language $\mathcal{L}$.

Definition 18 (Semantics of basic propositions)**.** *Let $\mathcal{M} = (W, R, I, AL)$ be a model for $\mathcal{L}$ and $w \in W$.*

$$\mathcal{M}, w \vDash \mathsf{has}_i m \;\Leftrightarrow\; m \in I_{w,i}$$

$$\mathcal{M}, w \vDash \mathsf{const}_i m \;\Leftrightarrow\; m \in \overline{I_{w,i}}$$

$$\mathcal{M}, w \vDash mk(\sigma) \;\Leftrightarrow\; \sigma \in AL(w)$$

The semantics for has *states that an agent possesses the message* m *at world* w *iff* m *is a member of his information state at world* w*. The semantics for* const *states that an agent* i *is able to construct a message* m *iff* m *can be derived from the information state of* i *at world* w *where* $\overline{I_{w,i}}$ *is the set of constructable messages from the set* $I_{w,i}$*. The proposition* $mk(\sigma)$ *holds at world* w *iff* σ *is marked at world* w*.*

Proposition 1. *Let* $m \in M$ *be a message and* $\mathcal{M} = (W, R, I, AL)$ *be a model,* $\mathbb{C}$ *be a set of construction rules such that* $m \vdash m \in \mathbb{C}$*. For all agents* $i \in A$ *it holds that* $\mathcal{M}, w \vDash \mathsf{has}_i m \to \mathsf{const}_i m$*.*

Proof. Assume $\mathcal{M}, w \vDash \mathsf{has}_i m$ holds for some agent $i \in A$. Then $m \in I_{w,i}$ by Definition 18, and by the construction rules of the model we have $m \vdash m$, thus by the closure of $I_{w,i}$ under $\mathbb{C}$ we have $m \in \overline{I_{w,i}}$. So, $\mathcal{M}, w \vDash \mathsf{const}_i m$. □

Remark 2. *It follows directly from Proposition 1 and contraposition that* $\mathcal{M}, w \vDash \neg\mathsf{const}_i m \to \neg\mathsf{has}_i m$*.*

As in Baltag et al. [1998], we define our action models with precondition and postcondition functions, which assign to each action label a proposition and substitution, respectively, which allow us to transform a given model after an update. These structures require each action to have a precondition and a pair of postconditions, where the precondition states which worlds the action can be executed in. The postconditions of the action act as a substitution function which assigns new valuations of propositions to the world where the action was executed and as a labelling mechanism to denote which actions have been executed in a given world. The preconditions will be propositions such that if the precondition is satisfied in the world then the action can be executed in the world. We define the substitution function $\Phi^A \to \mathcal{L}$ as in Definition 5. The set of all substitutions for Φ^A in $\mathcal{L}$ is $\mathsf{SUB}(\mathcal{L})$.

Definition 19 (Action Model)**.** *Define the action model as a tuple* $\mathfrak{A} = (\mathcal{A}, \sim, \mathsf{Pre}, \mathsf{Pos}_I, \mathsf{Pos}_{AL})$*. Let* $\mathcal{A}$ *be a finite set of actions* σ*. The function* $\sim\colon A \to \wp(\mathcal{A} \times \mathcal{A})$ *assigns each agent to a set of pairs of actions which represent the agents indistinguishability of events. We can refer to* $\sim$ *as a relation:* $\sigma \sim_i \sigma'$ *if* i *cannot distinguish between the execution of action* σ *and* σ'*. The relation* $\sim$ *is reflexive, symmetrical, and transitive. Let* Pre*,* Pos_I*, and* Pos_{AL} *be functions over the set of actions* $\mathcal{A}$ *which assigns to each action a precondition, a postcondition pertaining to information sets, and a postcondition pertaining to marked actions, respectively.* $\mathsf{Pre} : \mathcal{A} \to \mathcal{L}$ *is a formula in the language that denotes the precondition of* $\sigma \in \mathcal{A}$*.* $\mathsf{Pos}_I : \mathcal{A} \to \mathsf{SUB}(\mathcal{L})$ *is a function that assigns to all actions* $\sigma \in \mathcal{A}$ *a substitution for* $\mathcal{L}$ *such that* $Pos_I(\sigma)(p) \in \{\top, \bot\}$ *for all* $p \in \Phi_I$ *and* $Pos_I(\sigma)(p) = p$ *otherwise.* $Pos_{AL} : \mathcal{A} \to \mathsf{SUB}(\mathcal{L})$ *assigns to each* $\sigma \in \mathcal{A}$ *a substitution for* $\mathcal{L}$ *such that* $Pos_{AL}(\sigma)(p) \in \{\top, \bot\}$ *for all* $p \in \Phi_{AL}$ *and* $Pos_{AL}(\sigma)(p) = p$ *otherwise.*

Later in protocol descriptions, we will write $m \in I_{w,i}$ for the substitution that maps $\mathsf{has}_i m$ to $\top$ while leaving all other basic propositions unchanged. The effect of this postcondition adds m to the information set of i after the execution of the action. Similarly, σ^+ denotes the mapping

of $mk(\sigma)$ to $\top$ while leaving all other propositions unchanged, and σ^- denotes the mapping of $mk(\sigma)$ to $\bot$ while leaving all other propositions unchanged.[1]

Definition 20 (Model Update)**.** *Let σ, λ be actions in $\mathcal{A}$ and m be a message in M. Given a model $\mathcal{M}$ for $\mathcal{L}$ and an action model $\mathfrak{A}$, the updated model $\mathcal{M} \circ \mathfrak{A}$ is the tuple (W', R', I', AL'), defined as follows:*

- $W' = \{\langle w, \sigma\rangle \mid \mathcal{M}, w \vDash \mathsf{Pre}(\sigma)\}$
- $R' := \{R'_i \mid i \in A\}$ *such that* $(\langle w, \sigma\rangle, \langle v, \sigma'\rangle) \in R'_i$ *iff* $\langle w, v\rangle \in R_i$ *and* $\sigma \sim_i \sigma'$.
- $I'(\langle w, \sigma\rangle, i) = \{m \mid \mathcal{M}, w \vDash \mathsf{Pos}_I(\sigma)(\mathsf{has}_i m)\}$
- $AL'(\langle w, \sigma\rangle) = \{\lambda \mid \mathcal{M}, w \vDash \mathsf{Pos}_{AL}(\sigma)(mk(\lambda))\}$

The model update transforms a given model such that, the worlds are defined as world-action pairs where an action is paired with any world that satisfies its precondition. The accessibility relations of the updated model are now relations over world-action pairs such that if an agent cannot distinguish between two actions and two worlds, then the agent cannot distinguish between the action and world pairs composed of those actions and worlds. The information states of the updated model assign to a world and agent the set of messages the agent has at that world after the execution of the action. The updated action labels assign to each world the set of actions that are marked after the execution of an action at that world.

Definition 21 (Semantics of $\mathcal{L}$)**.** *Let $\mathcal{M} = (W, R, I, AL)$ and w be a world in W. Define the semantics of the language as follows with the basic propositions defined in Definition 18:*

$\mathcal{M}, w \vDash \top \;\Leftrightarrow$ *always*

$\mathcal{M}, w \vDash \neg\varphi \;\Leftrightarrow\; \mathcal{M}, w \nvDash \varphi$

$\mathcal{M}, w \vDash \varphi \wedge \psi \;\Leftrightarrow\; \mathcal{M}, w \vDash \varphi$ *and* $\mathcal{M}, w \vDash \psi$

$\mathcal{M}, w \vDash K_i\varphi \;\Leftrightarrow\;$ *for all v if wR_iv, then* $\mathcal{M}, v \vDash \varphi$

$\mathcal{M}, w \vDash [\mathfrak{A}, \sigma]\varphi \;\Leftrightarrow\;$ *if* $\mathcal{M}, w \vDash \mathsf{Pre}(\sigma)$, *then* $\mathcal{M} \circ \mathfrak{A}, \langle w, \sigma\rangle \vDash \varphi$

$\mathcal{M}, w \vDash [\mathfrak{A}]^*\varphi \;\Leftrightarrow\;$ *for all* $n \in \mathbb{N} : \mathcal{M}, w \vDash [\mathfrak{A}]^n\varphi$

Let $[\![\varphi]\!]_{\mathcal{M}} = \{w \in W \mid \mathcal{M}, w \vDash \varphi\}$. *Recall that* $[\mathfrak{A}]\varphi := \bigwedge_{\sigma \in \mathfrak{A}} [\mathfrak{A}, \sigma]\varphi$.

Remark 3. *The following follow from Definition 21 and the abbreviation of $[\mathfrak{A}]\varphi$ in Definition 15:*

- $\mathcal{M}, w \vDash [\mathfrak{A}]\varphi \;\Leftrightarrow\;$ *for all* $\sigma \in \mathfrak{A} : \mathcal{M}, w \vDash [\mathfrak{A}, \sigma]\varphi$
- $\mathcal{M}, w \vDash \langle\mathfrak{A}\rangle\varphi \;\Leftrightarrow\;$ *there is a* $\sigma \in \mathfrak{A} : \mathcal{M}, w \vDash \langle\mathfrak{A}, \sigma\rangle\varphi$
- $\mathcal{M}, w \vDash \langle\mathfrak{A}\rangle^*\varphi \;\Leftrightarrow$ *there are* $\sigma_0, ..., \sigma_n \in \mathfrak{A} : \mathcal{M}, w \vDash \langle\mathfrak{A}, \sigma_0\rangle ... \langle\mathfrak{A}, \sigma_n\rangle\varphi$

With the logical syntax and semantics in place, we can now take cryptographic protocols and model them within the logic. The protocols in this chapter rely on the exchange of information resulting from the execution of actions. The following section will analyze cryptographic protocols and demonstrate how the logic is capable of modeling them as well as verifying security goals. The protocols will be constructed by designing the action model $\mathfrak{A}$ to be executable in a modular way, so we can analyze potential intruders of different strengths.

[1] An example of when this might be required is if the precondition of an action σ is that σ is unmarked, thus preventing repeated execution of σ with every model update.

3.2 Cryptographic Protocols

Cryptographic protocols are typically designed with respect to the security goals they are attempting to reach. Since protocols are multi-agent interactions, the order of the sequence of actions is important as well as the contents of the message. These details are dependent on the network environment as well as the limitations of the agents, with respect to their epistemic states. This section will recall back to the logical language to express how a cryptographic protocol can be modeled within the logic as well as how the security goals can be verified. The set of actions executable within the model must express certain properties with respect to information exchange, furthermore the concept of an intruder must be given its own potential actions which test the security of the protocol. Finally, we will illustrate some examples of cryptographic protocols as well as verification of their security goals.

3.2.1 Verification

The execution of a protocol is the execution of a sequence of actions. While some of the actions may take place within the information states of individual agents, such as computing and constructing messages, the interactive portion of the protocol is the pattern of messages exchanged between the agents. We will use the following notation to describe an instance of communication: $a \to b : m$ denotes that a sends the message m to b. This notation will allow for a proper protocol description which we can then map to actions within the logic. For a communication described as $a \to b : m$, m is a message of a fixed pattern, meaning that given a protocol description we can assign parameters to the form of a given message. The parameters that compose these patterns each have a fixed domain specified in the initial model. Any parameter that has a specification that ranges over the set of agents is called a role.

In a run of a protocol, the action patterns are instantiated by some θ, which maps parameters of the protocol to their domain. In the context of a given protocol, Θ refers to the set of all possible instantiations. We will assume the set of parameters and their corresponding domain are all finite, therefore Θ is finite.

The network environment with which the agents use to communicate is also important to establish because it determines how exactly information is moved from one agent to another and what possibilities there might be for an intruder to conduct an attack on the protocol. To simulate a communication channel, we use an input buffer and an output buffer, adapted from Baeten [2020], which we can represent by two special agents within the model. The agent In will act as the input buffer, which all agents can send messages to, and the agent Out will act as the output buffer which will deliver messages to agents. The interaction between In and Out will not be made explicit, however if a sends a message m to b, then a must send the message to In, and then Out sends the message to b. It is in this buffer environment that an intruder may exist, eavesdropping messages from In, and passing on fake messages to Out. It holds that $I_{In} \subseteq I_{Out}$.

Lastly, we discuss the verification model which takes the security goals of the protocol and checks whether they are valid in all instantiations of the protocol. Assuming we have designed an action model $\mathfrak{A}$ that properly represent the protocol description, we may also choose to insert intruder actions Int and we will let $\mathfrak{A}_{Int}$ denote the set of actions containing the protocol description and possible intruder actions. Verifying the security goals of the protocol requires using a model as described in Definition 16 and the action model $\mathfrak{A}_{Int}$. We formalize the verification as follows:

$$\mathcal{M} \vDash \bigwedge_{\theta \in \Theta} [\mathfrak{A}_{Int}]^* [\mathfrak{A}_{Int}, \sigma(\theta)] \varphi(\theta)$$

where $\mathcal{M}$ is a model as defined in Definition 16 which includes all assumptions made at the initial distribution and all information states for all agents. $\mathfrak{A}_{Int}$ denotes the action model including the intruder actions, and $\sigma(\theta)$ is an action in $\mathfrak{A}_{Int}$, which is instantiated by θ, typically $\sigma(\theta)$ is the last action of a protocol, thus ensuring that we are verifying certain properties after a proper execution of the protocol. φ is a formula in $\mathcal{L}$, which we design to represent the security goal. This formula is also instantiated by θ, which means that any schematic parameters set in the protocol are mapped to a specific member of the domain. $\bigwedge_{\theta\in\Theta}[\mathfrak{A}_{Int}]^*[\mathfrak{A}_{Int},\sigma(\theta)]\varphi(\theta)$ is a formula that takes the conjunction over all possible instantiations of the parameters and says that after any possible execution of the protocol described by $\mathfrak{A}_{Int}$, φ holds. Given a φ which accurately represents the security goal, then the validity of the above formula verifies that the security goal holds within the model with the initial assumptions.

3.2.2 Modeling Actions

The first step to modeling a cryptographic protocol is to define the set of actions which are possible. After the actions are designed, the initial model can be set which contains all initial assumptions and parameters for each agent in the protocol. Let A be the set of agents in the protocol, this includes an adversarial agent. Given an instantiation of the protocol, one of the agents in A is mapped to the role of Eve, who is the intruder capable of performing intruder actions as listed in Table 3.2. Recall our model requires two special agents to represent the input and output buffer of the network environment, thus our agents will be $A \cup \{In, Out\}$.

Since a protocol description is composed of actions of the form $a \to b : m$, where a sends b the message m, we split this action into two parts to correspond with the network environment of the model. The *sending* part of the action is denoted by the action σ where $a \to In : m$ takes place. The *receiving* part of the action is denoted by the action δ where $Out \to b : m$ takes place. Since actions in $\mathfrak{A}$ are defined by the functions $\mathsf{Pre}, \mathsf{Pos}_I$, and Pos_{AL}, they need to be defined for each action. The precondition of the sending action is that the agent can construct the message they are sending from their information set, $\mathsf{Pre}(\sigma) = \mathsf{const}_a m$. Furthermore, to prevent a continuous execution of the same action over multiple updates, we stipulate that the action σ must be unmarked at a world where it is executed, formally this is denoted as $\neg mk(\sigma)$. The postcondition of the sending action is that the message is now in the information state of the input buffer, $\mathsf{Pos}_I(\sigma) = m \in I_{In}$. The action is then marked, so $\mathsf{Pos}_{AL}(\sigma) = \sigma^+$.

Similarly, for the receiving portion of the action we define $\mathsf{Pre}(\delta) = \mathsf{has}_{Out} m$, since the buffer agents are merely environmental they cannot construct messages from their information sets. Recall, In and Out are assumed to be network buffers that communicate amongst themselves, yet that communication is not made explicit within the models. The postcondition of δ is defined as $\mathsf{Pos}_I(\delta) = m \in I_b$. Lastly, the action label is defined $\mathsf{Pos}_{AL}(\delta) = \{\delta^+, \sigma^-\}$ signifying the *completion* of the message communication, which allows the σ action to be executed again. Table 3.1 lists the described actions with the corresponding pre-and-post-conditions. It should be noted that the preconditions and postconditions listed in the table are considered the most general and basic of message communication, but if one wanted to design more stringent rules on the interaction between the agents, more complex preconditions could be implemented in the form of complex formulas in $\mathcal{L}$.

To design an adversary's potential actions, the strength of the intruder must be taken into account. The strength must also be considered when interpreting the results of a verification formula, for instance if the intruder only has passive actions, then the security goals can only be said to hold in the presence of a passive adversary. For basic eavesdropping actions, which are considered to be passive actions, the intruder is allowed to eavesdrop messages from the

Action	Direction	Message	Pre	Pos_I	Pos_{AL}
σ	$a \to In$	m	$\mathsf{const}_a m \wedge \neg mk(\sigma)$	$m \in I_{In}$	σ^+
δ	$Out \to b$	m	$\mathsf{has}_{Out} m$	$m \in I_b$	δ^+, σ^-

Table 3.1: Basic sending and receiving actions

Action	Direction	Message	Pre	Pos_I	Pos_{AL}
take	$In \to Eve$	m	$\mathsf{has}_{In} m$	$m \in I_{Eve}$	-
fake	$Eve \to Out$	m	$\mathsf{const}_{Eve} m$	$m \in I_{Out}$	-

Table 3.2: Intruder action model Int

input buffer as well as insert messages to the output buffer.[2] For some arbitrary message m, this consists of two sub actions take and fake, where take is of the form $In \to Eve : m$ and fake is of the form $Eve \to Out : m$. With these two actions, the intruder can take any message from the input buffer and if the intruder can construct a message, then she can send that message via the output buffer. The precondition and postcondition of these actions are defined the obvious way and can be seen in Table 3.2. It should be noted, if one wanted a more powerful adversary within the model then it is easy to design more complex actions as well as implementing the adversary with more information in the initial state. An example of more aggressive adversarial actions not considered in this thesis, is a *jam* action which prevents a message from passing from the input buffer to the output buffer without the intruder learning the message.

The action model $\mathfrak{A}$ also possesses the relation $\sim$ which assigns epistemic properties to each action, namely whether an agent can distinguish the execution of one action from another. The general method for modeling actions and indistinguishability is to assume each agent in $i \in A$ can only distinguish those actions which either directly affect his information set or that he executed. That is, all other actions λ and ν have the relation $\lambda \sim_i \nu$. For simplicity, we can determine partitions on the set of actions $\mathcal{A}$ for each agent $i \in A$ with respect to a given action model $\mathfrak{A}$ which determines the actions they can distinguish. For example, the action model represented in Table 3.1 has the equivalence relations $\sim_a = \{\langle \sigma, \sigma \rangle\} \cup \{\langle \delta, \delta \rangle\}$ and $\sim_b = \{\langle \sigma, \sigma \rangle\} \cup \{\langle \delta, \delta \rangle\}$ since only the σ action is relevant to a and only the δ action is relevant to b. This example, however, represents a unique case in which there are only two action types: σ and δ. We suppose a general common knowledge over the set of possible action types, that is all agents know the actions as described in Table 3.1. We maintain that each agent can only distinguish those actions which are directly relevant to him, however in a case with only two types of actions, if an action is executed and the action is not relevant to an agent i, i will still recognize that an action took place and conclude that the executed action was of the other type.

With the set of basic actions defined, the initial model of a protocol can be set. The initial model $\mathcal{M} = (W, R, I, AL)$ is defined as follows:

> W: Define each world by the possible distribution of messages over the set of agents in A. This then determines all information sets in I so for a given distribution of the messages over the set A, $m \in I_{w,i}$ for all $i \in A$ such that $m \in M$ is distributed to i in world

[2]Standard notions in cryptography tend to state that a passive adversary can only read messages without the possibility of sending her own, however for our purposes the messages which Eve can send are restricted to those messages which she can construct. Therefore, at initial stages of a protocol, the adversary is only capable of reading messages, and then as she gains information over the course of the protocol, she can construct fake messages.

w. We set the input and output buffer information sets to the empty set for all worlds: $I_{w,In} = I_{w,Out} = \emptyset$.

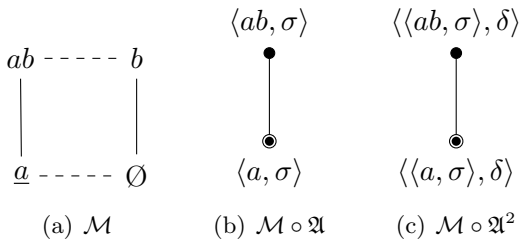
R_i: let $\langle w, v\rangle \in R_i$ iff $I_{w,i} = I_{v,i}$

AL: In the initial model, $AL(w) = \emptyset$ in all worlds $w \in W$.

Consider a basic action model $\mathfrak{A}$ that contains the actions of sending a message to the input buffer and receiving an action from the output buffer, as previously described by Table 3.1. Let $\mathcal{M}$ be a model with agents $\{a, b\}$ and a message m. There are four possible distributions of the message m amongst the set of agents including the distribution where neither of the agents possesses the message. Assuming that each agent only knows the information which they hold we can present a Kripke model $\mathcal{M}$ as follows:

- $W = \{a, b, ab, \emptyset\}$
- $R_a = \{\langle a, ab\rangle, \langle ab, a\rangle, \langle b, \emptyset\rangle, \langle \emptyset, b\rangle, \langle a, a\rangle, \langle b, b\rangle, \langle ab, ab\rangle, \langle \emptyset, \emptyset\rangle\}$
- $R_b = \{\langle b, ab\rangle, \langle ab, b\rangle, \langle a, \emptyset\rangle, \langle \emptyset, a\rangle, \langle a, a\rangle, \langle b, b\rangle, \langle ab, ab\rangle, \langle \emptyset, \emptyset\rangle\}$
- $AL(w) = \emptyset$ for all $w \in W$

Note that each world w is labeled by the name of the agent i such that $m \in I_{w,i}$ for each agent $i \in A$, so world ab is the world w in which $m \in I_{w,a}$ and $m \in I_{w,b}$. The accessibility relation R is symmetric, reflexive, and transitive. Figure 3.1(a) illustrates the Kripke model showing that each agent can only distinguish the worlds relevant to the information they hold. The solid lines denote the indistinguishability of a and the dashed lines join the worlds which b cannot distinguish. The actual world is underlined as $\underline{a}$, so b cannot distinguish the actual world from the world in which neither agent has the message m, and a cannot distinguish the actual world from the world where both a and b have the message m. Recall, the information sets of both the input and output buffer are the null set at this stage in the protocol.



(a) $\mathcal{M}$ (b) $\mathcal{M} \circ \mathfrak{A}$ (c) $\mathcal{M} \circ \mathfrak{A}^2$

Figure 3.1: Results of updating a model $\mathcal{M}$ with the action model $\mathfrak{A}$

The action model $\mathfrak{A}$ contains the actions σ and δ. The distinguishability of events in $\mathfrak{A}$ is such that a can only distinguish the σ action from all other actions, and b can only distinguish the δ action from all other actions. Notably, the $\sim_i$ relation is reflexive, so for all actions λ and agents i, $\lambda \sim_i \lambda$. Therefore when we update the model $\mathcal{M}$ with the action model $\mathfrak{A}$, the result is $\mathcal{M} \circ \mathfrak{A}$ as depicted in Figure 3.1(b), where the model is reduced to just those worlds where the precondition of σ holds. The δ action cannot be executed in the initial model, since by assumption $I_{In} = I_{Out} = \emptyset$ and thus the precondition of δ fails to hold in any world of the initial model.

Once more, the model in Figure 3.1(b) can be updated by the action model $\mathfrak{A}$ resulting in the model depicted in Figure 3.1(c). Although not explicitly illustrated in the model, the input

and output buffer both possess the message m since $I_{In} \subseteq I_{Out}$, thus satisfying the precondition of δ, allowing the action to be executed. Since the precondition of σ states that σ cannot be marked in the world in which it will be executed, the execution of σ cannot be repeated in this model until σ is unmarked. After the δ action is executed in Figure 3.1(c), b possesses the message m, thus changing the information set of an agent. At both worlds, it holds that $\mathsf{has}_a m \wedge \mathsf{has}_b m$, however the worlds are separate with respect to their histories which denote how the world was in the initial model.

The following subsection examines specific cryptographic protocol examples and demonstrates how to model them as well as verify specific security goals. These protocol examples will utilize potential intruder actions and encryption methods.

3.2.3 Protocol Examples

To introduce the concept of modeling cryptographic protocols, we will examine a very basic protocol where the agents have a shared key established in the initial state. Recall, the protocol description will apply to roles, so different instantiations can map any agent from the domain into a role, therefore our assumptions about what information is known in the initial distribution must be made with respect to the roles and not the agent names. For example in the following protocol between a sender S and a receiver R, we can map any of the agents from $A = \{a, b, c\}$ into the roles of S, R, and Eve by using different instantiations $\theta \in \Theta$. This first protocol is designed with privacy from eavesdroppers as the goal. First, a protocol description is necessary before implementing the initial state. This protocol model will also assume a passive adversary as described in Table 3.2.

3.2.3.1 Secrecy with a Shared Key

This protocol only requires one key, but it must be known to all agents with whom private communication wants to be shared. Let two principals share a key k that they use to encrypt messages. Since both agents have the key, they can send encrypted messages over an open channel and the other agent can decrypt the message without any eavesdropper obtaining the contents of the message. So, for the shared key k, as long as a message is of the form m_k, the two principals can confidently send messages to one another.

Since the important aspect of this protocol functioning successfully is the shared key, it requires the assumption that the two parties communicating, a sender and a receiver, both have a shared key only known to them. Let the set of agents be $A = \{a, b, c\}$ where Alice is a, Bob is b, and Charlie is c. The total set of agents in the model will be $A \cup \{In, Out\}$ and the roles of the protocol will be S for sender, R for receiver, and Eve for an eavesdropping intruder.

The protocol description only requires one message to be sent from S to R, and that is the encrypted secret they wish to share. Thus, using our formal notation we have:

1. $S \to R : m_k$

where m is the secret possessed by the sender, and m_k is the encrypted form of the message. As previously mentioned, the parameters are m, S, R where the domain of m is the finite set of secrets $\mathcal{S}$, and the domain of the roles is A. We do not list the key k as a parameter since it is fixed in this protocol, and thus remains the same in any instantiation. Since we assume the key is only known to the sender and receiver, they are both capable of sending encrypted messages over an open channel without an intruder obtaining their secrets, thus the protocol ensures secrecy through the shared key.

With the protocol description above, the action model can be designed to accurately represent the required action patterns. Since our model uses a network environment with input and

Action	**Direction**	**Message**	Pre	Pos_I	$\mathsf{Pos}_A L$
σ	$S \to In$	m_k	$\mathsf{has}_S m \wedge \mathsf{const}_S m_k \wedge \neg mk(\sigma)$	$m_k \in I_{In}$	σ^+
take	$In \to Eve$	X_k	$\mathsf{has}_{In} X_k$	$X_k \in I_{Eve}$	-
fake	$Eve \to Out$	X	$\mathsf{const}_{Eve} X$	$X \in I_{Out}$	-
δ	$Out \to R$	X_k	$\mathsf{has}_{Out} X_k$	$X_k \in I_R$	δ^+, σ^-

Table 3.3: The action model $\mathfrak{A}_{Int}$ for a shared key protocol

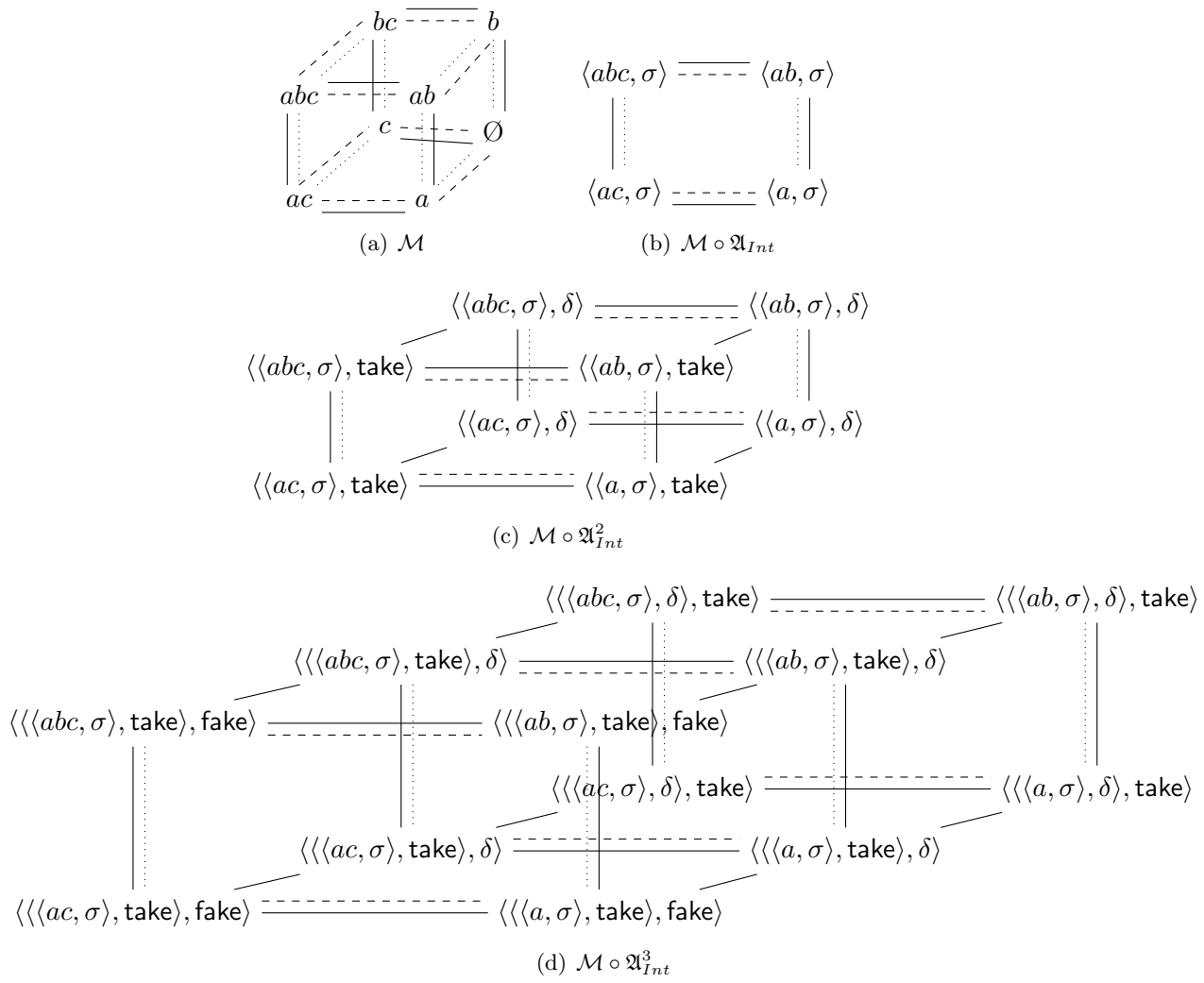
output buffers, the actions in the model $\mathfrak{A}$ reflect that. Table 3.3 lists the set of action patterns in $\mathfrak{A}$ with the addition of intruder actions Int, where any instantiation of these actions is an instance of the protocol described. As can be seen, the intruder's eavesdropper actions allow her to take the message m_k from the input buffer, however since k is not a part of Eve's information state, she cannot obtain the plaintext message m from m_k. The variable X is used to represent the parameters that cannot be controlled by the role performing the action, so when S sends the message m_k, they control the secret that is m, but when Eve takes the message from the input buffer, it need not be a secret message from $\mathcal{S}$. We do assume the action model is well-typed such that X will always be a message, but the content of the message is only determined by the parameter mapped to $\mathcal{S}$ in the σ action, in all other actions X can range over any message in M. This also ensures that there are only a finite amount of actions to be performed since the domains of the parameters of the protocol are finite, for example since there are a finite number of secrets in $\mathcal{S}$, then there are a finite number of σ actions, which implies there are also a finite number of take and δ actions since the σ actions determine what messages are in the input buffer. Furthermore, since there are only a finite number of take actions, then the information set of the intruder is finite implying that there are only a finite number of messages the intruder can construct for fake actions.

The equivalence relation for each agent over the set of actions is determined by whether the action changes the information set of the agent or whether the agent is executing the action. Therefore, the sender's equivalence relation can be denoted by the partitions $\sim_S = \{\langle\lambda,\nu\rangle | \lambda,\nu \in \{\sigma(\theta) \mid \forall\theta \in \Theta\}\} \cup \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\mathsf{take}(\theta), \mathsf{fake}(\theta), \delta(\theta) \mid \forall\theta \in \Theta\}\}$ which represents the sender's ability to distinguish σ actions, but no other action in $\mathfrak{A}_{Int}$. The instantiation θ in each partition represents the total set of actions given an action's parameters. The receiver's equivalence relation over the actions is denoted as $\sim_R = \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\delta(\theta) \mid \theta \in \Theta\}\} \cup \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\sigma(\theta), \mathsf{take}(\theta), \mathsf{fake}(\theta) \mid \forall\theta \in \Theta\}\}$, and the intruder's equivalence relation over the actions is $\sim_{Eve} = \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\mathsf{take}(\theta) \mid \forall\theta \in \Theta\}\} \cup \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\mathsf{fake}(\theta) \mid \theta \in \Theta\}\} \cup \{\langle\lambda,\nu\rangle \mid \lambda,\nu \in \{\sigma(\theta), \delta(\theta) | \forall\theta \in \Theta\}\}$. The receiver's equivalence relation is similar to the sender's such that the receiver can only distinguish one action, δ, and all other actions in $\mathfrak{A}_{Int}$ are indisitinguishable from one another. The intruder's equivalence relation contains three partitions since the intruder is capable of executing two actions: take and fake.

We will design the initial model with the simplest amount of information possible. The set of secrets $\mathcal{S} = \{m\}$ contains only the message m, and as previously mentioned only the roles of sender and receiver have the encryption key k. Thus for all agents $i \in A$, $I_{w,i} \subseteq \{k, m\}$. We will define the worlds of the initial model by the distribution of the secret message m. The name of a world will be denoted by the set of agents i such that $m \in I_{w,i}$.

Figure 3.2(a) illustrates the initial model[3] with the three agents a, b, c assigned by θ to the roles of S, R, Eve, respectively. The solid lines represent a's indistinguishability relation between worlds, the dashed line represents b's indistinguishability relations, and the dotted line denotes

[3]A similar illustration is found in Dechesne and Wang [2007] for the initial model of a cryptographic protocol, however it is missing some *distinguishability* relations. This is fixed in Figure 3.2(a) by using *in*distinguishability relations.



(a) $\mathcal{M}$ (b) $\mathcal{M} \circ \mathfrak{A}_{Int}$

(c) $\mathcal{M} \circ \mathfrak{A}^2_{Int}$

(d) $\mathcal{M} \circ \mathfrak{A}^3_{Int}$

Figure 3.2: Secret Sharing Scheme Model Updates

c's indistinguishability relations. Intuitively, each agent can only distinguish between the worlds that vary with respect to their own information set. Since there is only one message in the set $\mathcal{S}$, we can denote the secret message as m. The assumption that $I_{In} = I_{Out} = \emptyset$ in all worlds holds in the initial model. We can assign the world labelled a in the model of Figure 3.2(a) to be the actual world since a is the sender and is the only agent in A that possesses the secret message m. Formally we can say, $\mathcal{M}, a \vDash \mathsf{has}_a m \wedge \neg \mathsf{has}_b m \wedge \neg \mathsf{has}_c m$.

By updating the initial model with the action model $\mathfrak{A}_{Int}$, we obtain the resulting model in Figure 3.2(b). Only the worlds that satisfy the precondition of σ remain in the model, and since the input and output buffer are empty, per the assumption of the initial model, no other action's precondition is satisfied. Notably, a has the universal accessibility relation over the worlds in $\mathcal{M} \circ \mathfrak{A}_{Int}$ since a cannot distinguish between the distributions because $has_a m$ holds in all worlds and because a cannot distinguish one σ action from another σ action.

Updating $\mathcal{M} \circ \mathfrak{A}_{Int}$ with $\mathfrak{A}_{Int}$ again results in the model illustrated in Figure 3.2(c) which increases the number of worlds since the preconditions of two actions are satisfied at all the worlds of $\mathcal{M} \circ \mathfrak{A}_{Int}$. Since the input buffer contains a message, namely m_k, c can perform the take action. The transaction is omitted, but we let Out obtain information from In, so the δ action can be executed which delivers m_k to b. Both of these actions are distinguishable by c

and b, but not by a, which results in a's indistinguishability relation holding over all worlds in $\mathcal{M} \circ \mathfrak{A}^2_{Int}$.

Updating the model for a third time results in the model $\mathcal{M} \circ \mathfrak{A}^3_{Int}$ which is illustrated in Figure 3.2(d). The model increases in size once more because the postconditions of the two actions possible in the model $\mathcal{M} \circ \mathfrak{A}^2_{Int}$ satisfy the preconditions of the two actions fake and δ. The fake action is only possible if c can construct a message she intends to insert into the output buffer. Since it is possible for c to take the message m_k in a subset of worlds in the model $\mathcal{M} \circ \mathfrak{A}^2_{Int}$, these aforementioned worlds satisfy the precondition of the fake action. It is also possible for c to take the message in the previous update, but then to not do anything further and instead the message continues on its delivery course through the output buffer, also satisfying the precondition of δ. This can be seen as the *middle* quadrant of the model in Figure 3.2(d). Lastly, in a sequence where the message m_k is delivered to b from the output buffer, the message is still in the information set of the input buffer, thus allowing c to perform a take action after the fact. There is another possible action that is satisfied after the execution of δ, and that is to execute σ again. However, since there is only one message in the set $\mathcal{S}$ this would be a superfluous addition that would only serve to obfuscate the relevant information of the model in Figure 3.2(d). All three quadrants in the model which denote possible action sequences are joined by a's accessibility relation since c can distinguish the actions of fake and take, and b can distinguish the δ action. There are further expansions upon the model by updating it further with $\mathfrak{A}^n_{Int}$ for some n, however these models would quickly become unreadable.

This simple protocol has one straightforward goal: secrecy such that no unintended agent obtains the message m. This goal must be formalised in the language of $\mathcal{L}$ so that the validity of the statement in the constructed model can be checked. For the following goal and propositions, let $\mathcal{M}$ be the initial model as described above.

Goal 1 (Privacy)**.** *Let k be a shared key between the roles of S and R, and m a message in $\mathcal{S}$. If an intruder does not know the shared key at the beginning of a protocol, then the intruder will not be able to obtain any secret message m.*

$$\mathcal{M} \vDash \bigwedge_{\theta \in \Theta} (\neg \mathsf{has}_{Eve} k \to [\mathfrak{A}_{Int}]^* \neg \mathsf{const}_{Eve} m(\theta))$$

In order to prove the validity of the formula in Goal 1 we must utilize the following propositions.

Proposition 2. [4] *For any message m and any agent i, if i can construct m, then he can still construct m after an execution of any sequence of actions in $\mathfrak{A}_{Int}$:*

$$\mathcal{M} \vDash \mathsf{const}_i m \to [\mathfrak{A}_{Int}]^* \mathsf{const}_i m$$

Proof. Assume $\mathcal{M}, w \vDash \mathsf{const}_i m$ holds, so $m \in \overline{I_{w,i}}$. Then by updating the model $\mathcal{M}$ with the action model $\mathfrak{A}_{Int}$ given in Table 3.3, we obtain $\mathcal{M} \circ \mathfrak{A}_{Int}$. By Definition 19, Pos_I is a function that assigns to each action a substitution for $\mathcal{L}$, and there is no action and proposition pair (σ, p) such that $\sigma \in \mathfrak{A}_{Int}, p \in \Phi_I$, and $\mathsf{Pos}_I(\sigma)(p) = \bot$. That is to say that no postcondition of $\mathfrak{A}_{Int}$ deletes information from an agent's information set. Therefore, for any action $\sigma \in \mathfrak{A}_{Int}$, $\mathcal{M} \circ \mathfrak{A}_{Int}, \langle w, \sigma \rangle \vDash \mathsf{const}_i m$. Thus, there is no sequence of actions that would result in an agent losing the ability to construct a message he could previously construct. □

[4] This proposition is presented in Dechesne and Wang [2007], and again in Chen and Deng [2020], without a proof. I have adjusted the notation to fit present purposes and provided the proof.

The following proposition pertains to how there is no way to deduce a plaintext message from the encrypted form of that message. The only way to obtain the plaintext from the encrypted message is to have the corresponding key to the encryption. Let T be a set of messages and $\overline{T}$ denote the closure of T under $\mathbb{C}$.

Proposition 3. *Let $\mathbb{C}$ be the set of construction rules given in Definition 17. For any set of messages T, some key k, and any secret message m, if $k \notin \overline{T}$ and $m \notin \overline{T}$, then $m \notin \overline{\{m_k\} \cup T}$ and $k \notin \overline{\{m_k\} \cup T}$.*

Proof. Assume $k \notin \overline{T}$ and $m \notin \overline{T}$. Suppose a set $\overline{\{m_k\} \cup T}$. In $\mathbb{C}$, the only rules which have a conclusion of m are of the form $(m, m') \vdash m$, $m \vdash m$, and $\{m_k, k\} \vdash m$.

For the case of $(m, m') \vdash m$, it requires that $(m, m') \in \overline{T}$. Since $\overline{T}$ is the closure of T under $\mathbb{C}$, then by rule $(m, m') \vdash m$, $m \in \overline{T}$ which contradicts the assumption.

For the case of $m \vdash m$, it requires that $m \in \overline{T}$ which contradicts the assumption. For the case of $\{m_k, k\} \vdash m$, $k \notin \overline{T}$ by assumption, and thus the conditions for this rule are not satisfied. To prove $k \notin \overline{\{m_k\} \cup T}$ repeat the above argument but let $m = k$. □

Proposition 4. *Goal 1: 'Privacy' is true in the initial model $\mathcal{M}$.*

Proof. [5] For the purposes of this proof we take an arbitrary instantiation $\theta \in \Theta$. Without loss of generality, let the agents a, b, c in the initial model $\mathcal{M}$ be assigned to the roles of S, R, Eve, respectively and the secret message m is mapped to the only element of $\mathcal{S}$. For a proof by contradiction assume there is a world w in $\mathcal{M}$ such that

$$\mathcal{M}, w \not\models \neg \mathsf{has}_{Eve} k \rightarrow [\mathfrak{A}_{Int}]^* \neg \mathsf{const}_{Eve} m$$

It follows that there is a sequence of actions $\sigma_1, ..., \sigma_n \in \mathfrak{A}_{Int}$ such that

$$\mathcal{M}, w \models \neg \mathsf{has}_{Eve} k \wedge \langle \mathfrak{A}_{Int}, \sigma_1 \rangle ... \langle \mathfrak{A}_{Int}, \sigma_n \rangle \mathsf{const}_{Eve} m$$

Based on the initial model and assumptions on the information set of Eve, Eve does not have the encryption key, so for all worlds w it holds that $\mathcal{M}, w \models \neg \mathsf{has}_{Eve} k$. It follows from this assumption that in the initial model:

$$\mathcal{M}, w \models \neg \mathsf{const}_{Eve} k \wedge \langle \mathfrak{A}_{Int}, \sigma_1 \rangle ... \langle \mathfrak{A}_{Int}, \sigma_n \rangle \mathsf{const}_{Eve} m$$

since in the initial model the information set of Eve is empty. Let I_i^l be the information set of an agent i after executing actions $\sigma_1, ..., \sigma_l$ in $\mathcal{M}, w$ where I_i^0 is the information set in the initial model. From Proposition 2 it follows that there must be a unique l such that $m \notin \overline{I_{Eve}^l}$ but $m \in \overline{I_{Eve}^{l+1}}$. Of the actions designed in $\mathfrak{A}_{Int}$, only the following type of action could add information to Eve's information set:

- $\sigma_{l+1} := \mathsf{take}$

In this case $\overline{I_{Eve}^{l+1}} = \overline{\{m\} \cup I_{Eve}^l}$. The precondition of take is $m \in I_{In}$. The input buffer only contains messages which agents send and there is no σ action in $\mathfrak{A}_{Int}$ where the content of the message is not encrypted by the key k, so the information gained in σ_{l+1} must be of the form m_k. However, by Proposition 3 it follows that to deduce m from m_k one must have the key k, but this is in contradiction with the first conjunct. □

[5] This proof is inspired by techniques used in Dechesne and Wang [2007, pg. 12] for the proof of **R2** in relation to a different cryptographic protocol.

3.2.3.2 Secret Sharing Scheme and Man-in-the-Middle Attack

The following protocol examines a secret sharing scheme over an insecure network using symmetric keys, and does not rely on the assumption that the communicating agents already share an encryption key. The protocol was the topic of Chen and Deng [2020], and the security goals of the protocol have already been verified under the assumption of a passive adversary. The passive adversary had the available actions of take and fake and was implemented with no information set at the initial state. The verification of the security goals under the passive adversary will be analyzed as well as making a modification to the initial model which results in a more *intelligent* adversary capable of conducting a man-in-the-middle attack.

To begin, the protocol description relies on the notion that encryption is a commutative process, so for two distinct keys k and t an encrypted message m denoted by $\{\{m\}_k\}_t$ can be decrypted in any order, thus $m_{kt} = m_{tk}$. Let the protocol interaction take place between roles S and R, for sender and receiver, with each role's key being k_s and k_r. For readability, denote the encryption of message m with key k_r as m_r. Assuming each agent has their own individual key only known to them, the protocol begins with the sender encrypting the secret message m with her own key k_s as m_s and sending it over the open network. Once the receiver has received the message, he encrypts the message again with his own key, k_r, so the message is now of the form m_{sr} and sends it back over the open channel. After the sender has received this new message, she decrypts the message using her own key and sends m_r back over the channel. Now the receiver can open the secret message with his own key and obtain m.

The protocol description can be formalized as:

1. $S \to R : m_s$
2. $R \to S : m_{sr}$
3. $S \to R : m_r$

The same network environment consisting of an input and output buffer can be utilized to represent the open communication channel. Thus the agents we want to analyze epistemically can be set as $A = \{a, b, c\}$ and the agents in the model are $A \cup \{In, Out\}$. The roles of the protocol are S, R, and Eve.

We will define the action model $\mathfrak{B}$ to represent the protocol and the potential intruder's action are modelled by Int. The set of actions $\mathfrak{B}_{Int}$ can be seen in Table 3.4. Here we continue the use of typed variables (X, Y) where X represents parameters that are of message type that cannot be controlled by the agent executing the action and Y represents parameters that are of key type that cannot be controlled by the agent. Some actions are also consolidated for simplicity, for example the action δ_1 demonstrates the output buffer sending a message of the form X_Y to the receiver R, but as this happens multiple times in the protocol we only need to list it once in Table 3.4 since none of the parameters change.

The equivalence relation over the actions in $\mathfrak{B}_{Int}$ for each agent (as assigned by their role) are given as follows:

- $\sim_S = \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\sigma_1(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\delta_2(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\sigma_3(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\delta_1(\theta), \sigma_2(\theta), \mathsf{take}(\theta), \mathsf{fake}(\theta) \mid \theta \in \Theta\}\}$
- $\sim_R = \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\delta_1(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\sigma_2(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\sigma_1(\theta), \delta_2(\theta), \sigma_3(\theta), \mathsf{take}(\theta), \mathsf{fake}(\theta) \mid \theta \in \Theta\}\}$
- $\sim_{Eve} = \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\mathsf{take}(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\mathsf{fake}(\theta) \mid \theta \in \Theta\}\} \cup \{\langle \lambda, \nu \rangle \mid \lambda, \nu \in \{\sigma_1(\theta), \sigma_2(\theta), \sigma_3(\theta), \delta_1(\theta), \delta_2(\theta) \mid \theta \in \Theta\}\}$

Action	Direction	Message	Pre	Pos_I	Pos_{AL}
σ_1	$S \to In$	m_s	$\mathsf{has}_S m \wedge \mathsf{const}_S m_s$	$m_s \in I_{In}$	σ_1^+
δ_1	$Out \to R$	X_Y	$\mathsf{has}_{Out} X_Y$	$X_Y \in I_R$	δ_1^+
σ_2	$R \to In$	X_r	$\mathsf{has}_R X \wedge \mathsf{const}_R X_r$	$X_r \in I_{In}$	δ_1^-, σ_2^+
δ_2	$Out \to S$	X_Y	$\mathsf{has}_{Out} X_Y$	$X_Y \in I_S$	δ_2^+
σ_3	$S \to In$	m_Y	$\mathsf{const}_S m_Y$	$m_Y \in I_{In}$	δ_2^-, σ_3^+
take	$In \to Eve$	X_Y	$\mathsf{has}_{In} X_Y$	$X_Y \in I_{Eve}$	-
fake	$Eve \to Out$	X	$\mathsf{const}_{Eve} X$	$X \in I_{Out}$	-

Table 3.4: Set of actions $\mathfrak{B}_{Int}$ for the Secret Sharing scheme

For our purposes of verification, the initial model of the protocol will only contain the necessary information. Our set of secrets $\mathcal{S} = \{m\}$ again, contains only the message m, and the roles S and R will both have their own encryption keys k_s and k_r. We define the information sets of all agents $A = \{a, b, c\}$ such that $I_{w,i} \subseteq \{k_s, k_r, m\}$. Once again, we can define the worlds of the initial model by the distribution of the secret message m. For any instantiation $\theta \in \Theta$, the actual world of the initial model will be the world w such that $\mathcal{M}, w \vDash \mathsf{has}_S m \wedge \neg\mathsf{has}_R m \wedge \neg\mathsf{has}_{Eve} m$.

Verification of the security goals of this protocol will follow as before[6], where the goals of secrecy is formalised with relevant alterations to the formulas.

Goal 2 (Secrecy)**.** *The intruder does not learn the secret message if she did not already have the secret message at the beginning of the protocol.*

$$\mathcal{M} \vDash \bigwedge_{\theta \in \Theta} (\neg\mathsf{has}_{Eve} m(\theta) \to [\mathfrak{B}_{Int}]^* \neg\mathsf{const}_{Eve} m(\theta))$$

Proposition 5. *Goal 2 is true in the model $\mathcal{M}$.*

Proof. Since our set of secret messages $\mathcal{S}$ only contains m, we can omit the θ instantiation and assume without loss of generality that agents a, b, c are mapped to the roles S, R, and Eve, respectively. We proceed with a proof by contradiction. Let w be a world in the model $\mathcal{M}$ such that $\mathcal{M}, w \nvDash \neg\mathsf{has}_{Eve} m \to [\mathfrak{B}_{Int}]^* \neg\mathsf{const}_{Eve} m$. It follows that:

$$\mathcal{M}, w \vDash \neg\mathsf{has}_{Eve} m \wedge \langle \mathfrak{B}_{Int} \rangle^* \mathsf{const}_{Eve} m$$

Which implies there is a sequence of actions such that:

$$\mathcal{M}, w \vDash \neg\mathsf{has}_{Eve} m \wedge \langle \mathfrak{B}_{Int}, \sigma_0 \rangle \cdots \langle \mathfrak{B}_{Int}, \sigma_n \rangle \mathsf{const}_{Eve} m$$

Based on the assumptions made in the initial model about the information set of Eve it follows that

$$\mathcal{M}, w \vDash \neg\mathsf{const}_{Eve} m \wedge \langle \mathfrak{B}_{Int}, \sigma_0 \rangle \cdots \langle \mathfrak{B}_{Int}, \sigma_n \rangle \mathsf{const}_{Eve} m$$

Using a similar method to the proof of Proposition 4, there must be an action σ_l such that $m \notin \overline{I^l_{Eve}}$ and $m \in \overline{I^{l+1}_{Eve}}$. In order for the intruder to receive information, there is only one type of action:

- $\sigma_{l+1} = \mathsf{take}$

[6]Goal 2 appears in Chen and Deng [2020] as well as Proposition 5 which prove that they hold. However, the proofs by Chen and Deng make epistemic statements about the agents in the model with the knowledge operator that have been adjusted here to be solely formal statements about possession and construction.

Subsequently, $\overline{I^{l+1}_{Eve}} = \overline{\{m\} \cup I^{l}_{Eve}}$. The precondition of take for the message m is that $m \in I_{In}$ however all σ type messages in $\mathfrak{B}_{Int}$ which have an agent send a message to the input buffer have three possible forms: m_s, X_r, and m_Y. All three of these require a key to access the contents of the message, which would violate the assumption made in the initial model that Eve has no key. Furthermore, any possibility that there is some action σ_j where $j < l$, such that Eve receives a key is impossible since once again, the messages will be of the form m_s, X_r, and m_Y and would require Eve to have a key. □

To modify the protocol described in Chen and Deng [2020] to make the adversary more intelligent at the initial model, consider a situation in which the intruder has her own encryption key k_{eve}, and upon intercepting a message m from the input buffer, she can now construct m_{eve} from her information set. Implementing the assumption that Eve has her own encryption key in the initial model allows the protocol to become susceptible to a man-in-the-middle attack. After the sender has sent the message m_s to the input buffer, Eve takes the message m_s and applies her own encryption key to it, constructing $m_{s,eve}$. Based on the modular format of $\mathfrak{B}_{Int}$, we only need to add one action to the action model $\mathfrak{B}_{Int}$ to make this possible. By adding the action of Table 3.5 to the action model depicted in Table 3.4, we can obtain the new action model $\mathfrak{B}^{+}_{Int}$. The construction of this action can be thought of as modifying one of the parameters of the fake action as described in Table 3.4. Namely, modifying the key parameter in the fake action such that Eve has control over the key used. Now, Eve can implement the fake message $m_{s,eve}$ in a fake action so the output buffer sends the sender $m_{s,eve}$. The sender continues on with the protocol, thus decrypting the message with her own key and sending m_{eve} to the input buffer. From here it is easy for Eve to use her own encryption key to obtain the secret message m.

The initial model $\mathcal{M}$ can be described as before with the additional modifying assumption that Eve has her own key k_{eve}.

Action	**Direction**	**Message**	Pre	Pos_I	Pos_{AL}
fake	$Eve \to Out$	X_{eve}	$\mathsf{const}_{Eve}X_{eve}$	$X_{eve} \in Out$	-

Table 3.5: Action for Man-in-the-Middle attack when adversary has k_{eve}

Goal 2 will prove not to hold with the action model of $\mathfrak{B}^{+}_{Int}$, since Eve will be able to construct the secret message m after the attack. Assuming the actual world is the world in the initial model $\mathcal{M}$ such that $\mathsf{has}_S m \wedge \neg\mathsf{has}_R m \wedge \neg\mathsf{has}_{Eve} m$ holds, it is easy to check based on the preconditions of each action that

$$\mathcal{M}, S \vDash \neg\mathsf{has}_{Eve} m \wedge \langle\sigma_1\rangle\langle\mathsf{take}\rangle\langle\mathsf{fake}\rangle\langle\delta_2\rangle\langle\sigma_3\rangle\langle\mathsf{take}\rangle\mathsf{const}_{Eve} m$$

holds, providing a sequence of actions that is a counterexample to Goal 2.

3.2.3.3 RSA Signature Scheme

With a slight adjustment to the construction rules regarding encryption, the logic becomes powerful enough to model the RSA digital signature protocol [Rivest et al., 1978]. By changing to a public-key encryption method, the language obtains two types of keys: public and private keys. Encrypting a message with a key works the same, so long as the conditions of having a key and having a message are met, one can construct an encrypted form of that message. However, for decrypting an encoded message it is no longer sufficient to have the key with which it was encrypted. Instead, one must have the corresponding *inverse* of the key. Intuitively, we can denote keys as k^{+} and k^{-} where k^{-} is the inverse of k^{+}, and vice versa. Given a message

m, $\{m\}_{k^+}$ denotes an encryption of the message with a public key, and $\{\{m\}_{k^+}\}_{k^-}$ denotes the private key applied to the encrypted message such that $\{\{m\}_{k^+}\}_{k^-} = m$. Another notable feature of public-key encryption is the commutativity of the keys, so $\{\{m\}_{k^+}\}_{k^-} = \{\{m\}_{k^-}\}_{k^+}$.

A network environment suitable for public-key encryption would require all trusted agents in the system to have a pair of keys, one private and one public. The public key of each agent would be known to everyone, or at least accessible via some database, and the private key of each agent is known only to the agent to whom the key belongs. Thus to send messages in the network, a sender only needs to know the public key of the person to whom they wish to send the message to and then encrypt the message with the recipient's public key. With the message encrypted, the sender can send the message over an open channel and the only agent who can decrypt and read the contents of the message is the agent with the corresponding private key, which is the intended recipient.

The following definition provides the new cryptographic construction rules for public-key encryption.

Definition 22 (Public-Key Encryption Construction Rules)**.** *The following rules determine what messages an agent can construct from his information set. Let $i \in A$ and this set of public-key construction rules will be referred to as $\mathbb{C}^+$.*

$$\frac{m \quad k_i^+}{\{m\}_{k_i^+}} \qquad \frac{m \quad k_i^-}{\{m\}_{k_i^-}} \qquad \frac{m_{k_i^+} \quad k_i^-}{m} \qquad \frac{m_{k_i^-} \quad k_i^+}{m} \qquad \frac{m \quad m'}{(m,m')} \qquad \frac{(m,m')}{m} \qquad \frac{m \quad m'}{m'} \qquad \frac{m}{m}$$

With the public-key encryption construction rules in place, it is now possible to construct a model that accurately represents the RSA signature scheme.

Digital signatures work similarly to how handwritten signatures work on personal letters in that they are a unique and *ideally* unforgable identifying mark on the letter that signifies to the reader that the sender is who they say they are. Rivest et al. [1978] designed a digital signature scheme that utilizes public-key encryption in order to generate personal signatures for each agent in a network. The signatures rely on the commutativity of the public and private keys when applied to a message and the fact that it is infeasible to derive a private key from a corresponding public key.

Consider two agents that want to exchange secret messages while also verifying that the recipient knows who created each message. Suppose Alice is the signer of the message and Bob is the verifier. Alice has the public-key encryption pair of (k_s^+, k_s^-) as well as all other public keys for all other agents and Bob has the public-key encryption pair of (k_v^+, k_v^-) as well as all other public keys for all other agents. The protocol works by Alice first decrypting a message m with her private key, obtaining $\{m\}_{k_s^-}$. Then, Alice sends this message to Bob by encrypting it with Bob's public key, so the message being sent over the open channel is of the form $\{\{m\}_{k_s^-}\}_{k_v^+}$. Even though the message is sent over an open channel, only Bob can obtain the contents of the message since he is the only one with the corresponding private key k_v^-. After decrypting the message, Bob has $\{m\}_{k_s^-}$ which means he cannot obtain the secret message m without first encrypting the message using Alice's public key. The private key decryption of the message m *is* the signature, since in order to view m, Bob must use Alice's public key and since Alice is the only agent in the network that possesses her private key, Bob is able to verify that only Alice could have constructed the signature.

Since the signature protocol utilizes individual agent computation to sign and to verify, the protocol description of sent messages is only an exchange of one message:

1. $S \to V : \{\{m\}_{k_s^-}\}_{k_v^+}$

Action	Direction	Message	Pre	Pos_I	Pos_{AL}
σ	$S \to In$	$\{\{m\}_{k_s^-}\}_{k_v^+}$	$\mathsf{has}_S m \wedge \mathsf{const}_S\{\{m\}_{k_s^-}\}_{k_v^+}$	$\{\{m\}_{k_s^-}\}_{k_v^+} \in I_{In}$	σ^+
take	$In \to Eve$	X_Y	$\mathsf{has}_{In} X_Y$	$X_Y \in I_{Eve}$	-
fake	$Eve \to Out$	X_Y	$\mathsf{const}_{Eve} X_Y$	$X_Y \in I_{Out}$	-
δ	$Out \to V$	X_Y	$\mathsf{has}_{Out} X_Y$	$X_Y \in I_V$	δ^+

Table 3.6: Action model $\mathfrak{C}_{Int}$ for the RSA Signature Scheme with passive intruder

where S is the signer of the message m and V is the verifier of the signature. Assuming the standard network environment and set of agents A assigned to the roles of S, V, Eve, we can build the action model $\mathfrak{C}_{Int}$ as seen in Table 3.6. The action model $\mathfrak{C}_{Int}$ contains the same equivalence relations for each agent as the action model $\mathfrak{A}_{Int}$ with the exception that the role R is substituted with the role V.

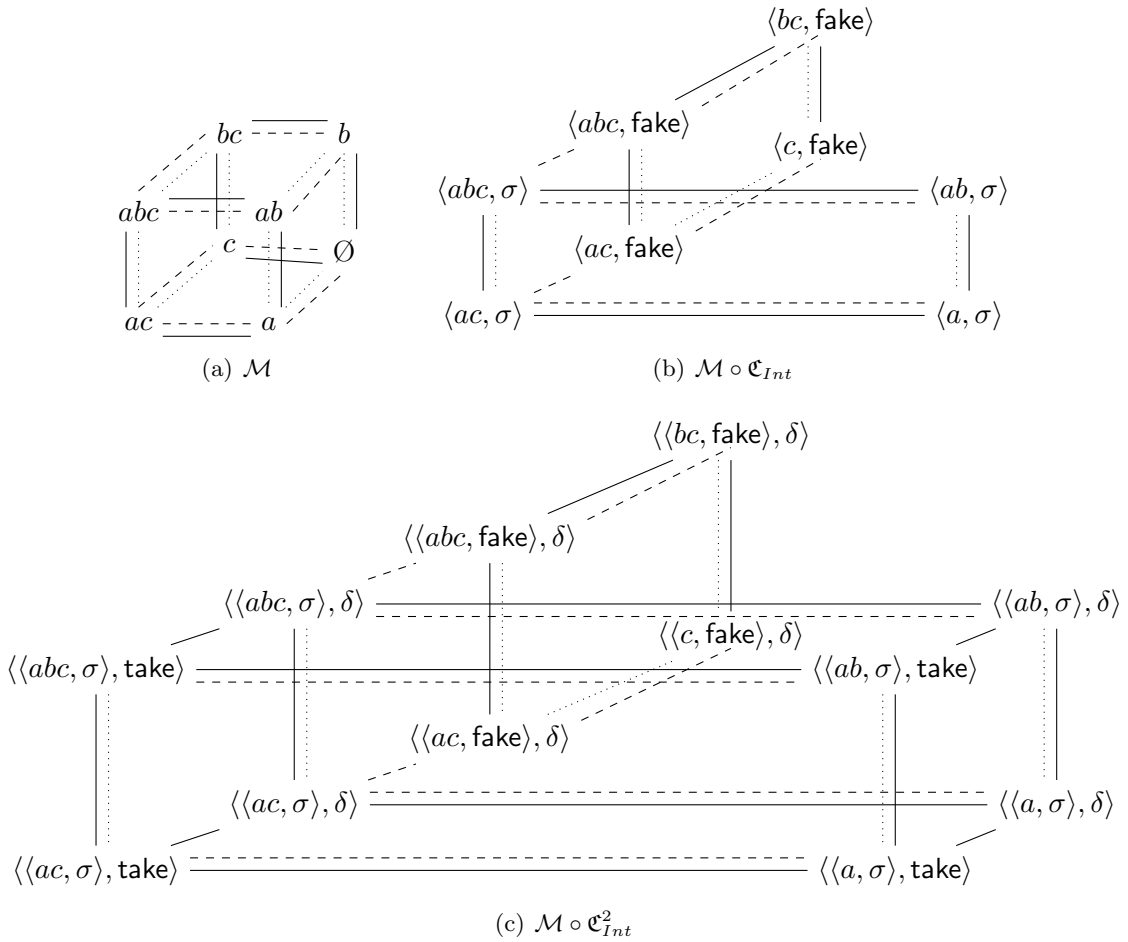
The initial model can be designed with the parameter for secret messages being the set of messages $\mathcal{S} = \{m\}$, and a set of worlds constructed as before in previous examples with a single secret message. The actual world will be the world that satisfies $\mathsf{has}_S m \wedge \neg\mathsf{has}_R m \wedge \neg\mathsf{has}_{Eve} m$. For all agents $i \in A$, the information sets are defined as $\{k_s^+, k_v^+, k_{eve}^+, k_{i(\theta)}^-\} \subseteq I_{w,i} \subseteq \{k_s^+, k_v^+, k_{eve}^+, k_s^-, k_v^-, k_{eve}^-\} \cup \mathcal{S}$. The initial distribution will ensure that all agents know the public keys of all other agents as well as their own private key, and the agent in the role of S has the secret message which they will sign.

The initial model of the RSA protocol contains more information than the initial model of the shared key example with respect to each agent's information set since each agent has at least the set of all public keys as well as their own private key. With respect to the set of secret messages $\mathcal{S}$, the information is distributed in the same manner as the shared key example and therefore the initial models are identical. An illustration of the initial model can be found in Figure 3.3(a) where we suppose the agents a, b, c are assigned to the roles S, V, Eve, respectively. This instantiation makes the world a the actual world based on our assumptions about the distribution of messages in the initial model.

Updating the initial model with $\mathfrak{C}_{Int}$ results in $\mathcal{M} \circ \mathfrak{C}_{Int}$, seen in Figure 3.3(b), which provides two possible actions to execute. Either the signer of the protocol can send a signed message, or the intruder can insert a fake message into the output buffer. The σ action can only be executed at those worlds where the signer has the message m, so all worlds w where $m \in I_{w,a}$, while the fake action can be executed from all worlds w where $m \in I_{w,c}$. Notably, these two sets of worlds intersect at the worlds abc and ac, resulting in four possible worlds with each world paired with one of the possible actions. In the model $\mathcal{M} \circ \mathfrak{C}_{Int}$, the new worlds $\langle abc, \sigma\rangle$ and $\langle abc, \mathsf{fake}\rangle$ are joined by the accessibility relation of b since b cannot distinguish between σ actions and fake actions, while the other agents can distinguish at least one. Similarly for the worlds $\langle ac, \sigma\rangle$ and $\langle ac, \mathsf{fake}\rangle$.

The model in Figure 3.3(c) illustrates updating $\mathcal{M} \circ \mathfrak{C}_{Int}$ with $\mathfrak{C}_{Int}$. From the quadrant of worlds where σ took place, two new actions become possible: δ and take because there is now a message in the buffer satisfying the preconditions of these actions. Similarly, in the quadrant of worlds where the fake action took place, it is possible for a δ action to occur, namely the delivery of the fake message constructed. The quadrant of worlds where the σ action is followed by the take action is joined to the quadrant of worlds satisfying the execution of σ followed by δ via a's equivalence relation. The actions δ and take are indistinguishable for a, while the other agents can distinguish at least one of the actions.

The security goals we want to show with this protocol are twofold: first, standard secrecy of the messages must hold such that no eavesdropper can obtain the secret message, and secondly the verifier must be able to verify the identity of the sender of the message. The formula



(a) $\mathcal{M}$ (b) $\mathcal{M} \circ \mathfrak{C}_{Int}$

(c) $\mathcal{M} \circ \mathfrak{C}^2_{Int}$

Figure 3.3: RSA Model Updates

for secrecy, can follow the standard secrecy formulas designed in previous protocol verification sections with only slight alterations to relevant propositions.

Goal 3 (RSA Secrecy)**.**

$$\mathcal{M} \vDash \bigwedge_{\theta \in \Theta} (\neg \mathsf{has}_{Eve} m(\theta) \to [\mathfrak{C}_{Int}]^* \neg \mathsf{const}_{Eve} m(\theta))$$

Goal 4 (RSA Verification)**.** *For any run of the protocol described in $\mathfrak{C}_{Int}$ ending with the action δ, the verifier knows that the signer constructed the signed message.*

$$\mathcal{M} \vDash \bigwedge_{\theta \in \Theta} ([\mathfrak{C}_{Int}]^* [\mathfrak{C}_{Int}, \delta(\theta)] \mathsf{const}_V \{m(\theta)\}_{k_S^-} \to K_V \mathsf{const}_S \{m(\theta)\}_{k_S^-})$$

Proposition 6. *Goal 3: 'RSA Secrecy' is true in the initial model $\mathcal{M}$.*

Proof. The proof can be constructed similarly to the secrecy proofs of the previous examples. The assumptions of the initial model and the patterns of the available actions in $\mathfrak{C}_{Int}$ imply that all messages that pass through the buffer must be encrypted. By the assumptions of the initial model, agents only have the private keys which belong to them, so therefore any message sent to the buffer cannot be read by an intruder *Eve* since *Eve* will not have the corresponding k_v^- key to decrypt the messages of a σ action. □

Proposition 7. *Goal 4: 'RSA Verification' is true in the initial model $\mathcal{M}$.*

Proof. Let w be an arbitrary world in the initial model $\mathcal{M}$ and m be a message instantiated by any $\theta \in \Theta$. We want to show that if $\mathcal{M}, w \vDash [\mathfrak{C}_{Int}]^*[\mathfrak{C}_{Int}, \delta]\mathsf{const}_V\{m\}_{k_S^-}$ holds then $\mathcal{M}, w \vDash K_V\mathsf{const}_S\{m\}_{k_S^-}$ holds. After updating $\mathcal{M}, w$ with $[\mathfrak{C}_{Int}]^*[\mathfrak{C}_{Int}, \delta]$, $\mathsf{const}_V\{m\}_{k_S^-}$ holds if the message sent in the δ action is of the form $\{\{m\}_{k_S^-}\}_{k_V^+}$. Thus, it only holds in those worlds where $\mathsf{has}_S m$ holds since $\{m\}_{k_S^-}$ can only be constructed by S. Furthermore, it only holds in those worlds where a σ action was executed containing $\{\{m\}_{k_S^-}\}_{k_V^+}$ since S is the only agent in the protocol that has the key k_S^- by assumption and therefore, the message $\{m\}_{k_S^-}$ could not have been faked by an intruder.

By the assumptions of a public-key encryption protocol, no agent has another agent's private key, therefore it is not possible that the verifier constructed the message $\{m\}_{k_S^-}$ himself, since it would require encrypting m with S's private key. Thus, in the worlds $\langle \cdots \langle w, \sigma_1 \rangle, ... \rangle, \delta \rangle$ in $\mathcal{M} \circ \mathfrak{C}^*_{Int}$ that satisfy $\mathsf{const}_V\{m\}_{k_S^-}$, the verifier had to receive the message $\{m\}_{k_S^-}$ after an action in the sequence $\sigma_1, ..., \sigma_n, \delta$.

In the model $\mathcal{M} \circ \mathfrak{C}^*_{Int}$, there are no possible worlds in which V receives $\{m\}_{k_S^-}$ and $\mathsf{const}_S\{m\}_{k_S^-}$ does not hold. Therefore, in all worlds accessible by V after receiving $\{m\}_{k_S^-}$, $\mathsf{const}_S\{m\}_{k_S^-}$ holds. So, in $\mathcal{M}, w$ if $[\mathfrak{C}_{Int}]^*[\mathfrak{C}_{Int}, \delta]\mathsf{const}_V\{m\}_{k_S^-}$ holds then $K_V\mathsf{const}_S\{m\}_{k_S^-}$ holds.

□

3.3 Review and Discussion

This chapter has presented the syntax and semantics of a cryptographic variant of DEL first introduced in Dechesne and Wang [2007], inspired by traditional Dolev-Yao models [Dolev and Yao, 1983]. The basic propositions of the logic represent whether an agent in the model is in possession of, or can construct, a message. This makes the model suitable for representing cryptographic protocols where each step in the protocol requires an exchange of messages amongst the agents. Furthermore, we examined the set of construction rules which makes cryptographic processes and deductions possible such as encryption and concatenation. Rather than designing mathematically sound encryption functions, we abstract away from such notions and assume our encryption operations work akin to one-way functions, such that they are easy and fast to generate and reversing the function is computationally infeasible. Therefore, having an encrypted message is not the same as having the message unless the agent in possession of the encrypted message also possesses the corresponding encryption key.

Using standard action dynamics and model updates [Baltag et al., 1998, van Ditmarsch et al., 2015], we could accurately model protocol descriptions within the logic. Upon updating given models with the protocol actions, it is possible to design sentences in the logical language which represent security goals of the protocol that can then be verified within the model. To give a few examples, we modeled two standard symmetric key protocols and the RSA public-key signature scheme. Within all of these examples under passive adversary assumptions, we could verify standard message secrecy which stated that no eavesdropper could obtain the secret message during or after the protocol execution. However, when modeling the secret sharing scheme presented in Chen and Deng [2020], if we adjusted the strength of the intruder, giving her the capability to conduct a man-in-the-middle attack, she could obtain the secret message.

The use of instantiations in the logic introduced by Dechesne and Wang allows a protocol to be re-usable in the sense that the actions in the protocol are not specific to particular messages or agents, only to the patterns which the parameters hold. The logic presented in this chapter

deviates from the original formulation by preventing interleaving instantiations which is allowed in Dechesne and Wang [2007] and Chen and Deng [2020]. The choice to prevent interleavings, which allow agents to take on multiple roles in a given instance of a protocol by using multiple instantiations simultaneously, is made to prevent undesirable deductions by the agents. The assumptions made in the initial model with respect to each agent's information set, was specified according to the role the agent was taking on in the protocol. For example, to allow an agent to be both a receiver and an intruder in a basic secret sharing scheme would allow an intruder to have a secret key under the initial assumption which would make breaking the scheme trivial. Ways around such a problem would require more stringent and dynamic initial assumptions that are less general than those presented in this chapter and directly pertain to a given initial model as well as the roles and strategies utilized by the agents.

In a real-world setting, symmetric key encryption is cheaper from a computational perspective than asymmetric key encryption since RSA requires larger encryption keys as factoring algorithms get more efficient, but symmetric key encryption relies on the assumption that all trusted agents already have a shared key, which is unrealistic [Hasib and Haque, 2008]. Therefore, it is common to implement a hybrid encryption method in which agents in a network perform a *key exchange* and authenticate identities using asymmetric encryption keys, like RSA. This does not require any common knowledge assumptions amongst the agents. Once identities have been authenticated and the agents have exchanged a symmetric key with one another, they begin communicating encrypted messages with the symmetric key.

More complex DEL variants have been constructed capable of modeling the Diffie-Hellman key exchange [Gattinger and van Eijck, 2015]. This variant required a heavier reliance on cryptographic primitives such as prime numbers and algebraic operations. In order to model epistemic relations over large sets of numbers, it requires a transition from Kripke models to register models [van Eijck and Gattinger, 2015] which encodes indistinguishability over countably infinite worlds into a finite set. Furthermore, rather than create a network environment such as input and output buffers for the messages of the protocol to pass through, attention-based announcements, introduced in van Ditmarsch et al. [2013], are utilized which assigns a set of agents that are *listening*. After a public announcement is made in the model, two copies are created, one in which those agents that are listening receive the information of the announcement and one in which the inattentive agents maintain the same level of ignorance.

While the above models and methods of implementation abstract away from the fundamental mathematics behind cryptography, they still provide a useful logical model that can reveal structural weaknesses in the protocol description. The work of Lowe [1996] famously revealed a flaw in the Needham-Schroeder protocol by analyzing a symbolic representation of the protocol.

CHAPTER 4

A Logic for Zero-Knowledge Protocols

4.1 Introduction

The next section of this thesis will focus on zero-knowledge interactive proofs and using DEL to verify whether certain epistemic properties hold of the protocol. In order to do so, cryptographic principles and techniques need to be explained so that these principles and techniques can be later introduced in a purely logical system. Then the characterization of knowledge in a cryptographic setting must be analyzed as the standard notion of knowledge in modal logic does not suffice for applications towards zero-knowledge [Halpern et al., 2009]. Using a formal characterization, we can extend the logic described in the previous chapter to be able to express cryptographic notions essential to zero-knowledge protocols. Thus, this will enable us to model zero-knowledge protocols and verify the security goals of the protocol.

There is existing literature on symbolic systems capable of modeling zero-knowledge protocols. However, these systems extend the standard Dolev-Yao models [Dolev and Yao, 1983] to capture a notion of indistinguishability within zero-knowledge proofs. Baskar et al. [2009] developed a type system which allows an interactive proof to be represented by a sequence of typed terms such that proofs are not communicated between agents, but instead the content of the proof is expressed by assigning the correct type to the appropriate term. Backes et al. [2007, 2015] implement a zero-knowledge constructor zk in the term algebras of Dolev-Yao models. These methods aim to verify security features of protocols through these symbolic systems yet they require complex and rigid syntax as well as constructing new derivation rules or a heavy reliance on computational soundness. Using DEL to model zero-knowledge protocols stems from a desire to incorporate the elegance of epistemic logics with the succinct semantics for model updates [Baltag and Moss, 2004].

A preliminary attempt was made to characterize zero-knowledge proofs in epistemic logic in Halpern et al. [1988] and was later refined in Halpern et al. [2009]. This required defining *practical knowledge* which expressed knowledge of a proposition with respect to what that proposition implied as well as defining temporal epistemic models in which the protocol runs could take place. Halpern et al. provide a formal sentence within their temporal epistemic logic that holds iff a protocol is a zero-knowledge protocol. Furthermore, the characterization used by Halpern et al. is effective at reasoning about the system and run of a given protocol, yet lacks a method for detailed analysis of specific interactions and communications made in a

given protocol. That is, one could not use the formal characterization as a means of reasoning about the epistemic states of the parties in an interaction.

Kramer [2007] designed a probabilistic polynomial-time symbolic system (PPL), in the spirit of Dolev-Yao models, aimed at reasoning about many types of cryptographic protocols. Once again this comes at the cost of a complex syntax and semantics, however the language itself is based in logical notions of knowledge and implication. We implement some logical techniques from PPL, such as the cryptographic parsing function, that makes modeling the protocols more accurate.

The following section will provide an overview for the cryptographic notions required for zero-knowledge as well as a brief explanation of the computation and probability necessary, before moving on to logical approaches to define these notions. This will be necessary for explaining how the interactions of a zero-knowledge proof ought to be perceived and how knowledge should be accounted for in the logic.

4.2 Zero-Knowledge Proofs

The central idea behind a zero-knowledge proof, at least from an epistemic perspective, is that the verifier in the interaction is not able to deduce any more information after the interaction than he was able to before the interaction, other than the validity of the proof. An interactive proof works by using two principals: a prover and a verifier, which can both be represented by an algorithm that determines the strategy they use based on an individual input and a common input shared between them. Let P and V be the algorithm used by the prover and verifier, respectively, with x being the common input shared between them. Thus an interactive proof can be described by the pair of algorithms (P, V) [Goldwasser et al., 1989]. Both P and V can do their own internal computations as well as send the other principal a value or input, which makes them interactive algorithms. An execution of (P, V) is a sequence of rounds such that for any round, V performs a computation and then outputs a value to P and then P performs a computation and outputs a value to V. In this manner, both P and V alternate making computations and sending information to one another. Each principal in the interactive proof can only see their own computations, their own auxiliary information, and a transcript of the exchanged messages between the principals. Finally, the execution ends when V either enters an *accepting* halt state or a *rejecting* halt state.

(P, V) is said to be an interactive proof system for a *language*[1] L if, when (P, V) is run on input x, then after the protocol execution, the verifier is convinced with high probability that $x \in L$. The prover and verifier of an interactive proof are considered *good*, when they follow the strategy imposed by the algorithms P and V, respectively. To denote the possibility that an algorithm is not good (ie. deviating from its prescribed strategy to attempt to manipulate the other party), we can refer to it as $\hat{P}$ or $\hat{V}$. To say that the verifier of an interactive proof for the language L is convinced that $x \in L$, means that on input x, (P, V) enters an accepting halt state. The run time of P and V during the execution of (P, V) is the number of rounds for each principal, respectively, and V is assumed to be limited by polynomial-probabilistic computations for each round, so V runs in time polynomial bounded by the length of the input, $|x|$. No assumptions about the run time of P are made at this point. The notion of being convinced with high probability is determined by a negligible function. $\epsilon : \mathbb{N} \to [0, 1]$ is negligible if for every positive integer z, there exists a $n_0 \in \mathbb{N}$ such that for all $n > n_0$: $\epsilon(n) < n^{-z}$. That is, for sufficiently large n, ϵ grows slower than any inverse polynomial [Katz and Lindell, 2007]. Thus, the verifier in an interactive proof is convinced with high probability

[1]The word 'language' here is defined differently than standard logical uses of 'language'. In this context, 'language' refers to the set of inputs for a given Turing machine that results in an accepting halt state.

if on input of a sufficiently large x, the probability that V enters an accepting halt state is greater than $1 - \epsilon(|x|)$.

Definition 23 (Interactive Proof Systems)**.** *Let L be a language over $\{0,1\}^*$ and let (P,V) be an interactive proof system. Let $\mathbb{P}((P,V)(x))$ be the probability that (P,V) enters an accepting halt state on input x. (P,V) is an interactive proof system for the language L if the following conditions are satisfied:*

- ***Completeness:*** *For a sufficiently large $|x|$, if $x \in L$, then $\mathbb{P}((P,V)(x)) \geq 1 - \epsilon(|x|)$*
- ***Soundness:*** *For a sufficiently large $|x|$ and any protocol $\hat{P}$, if $x \notin L$, then $\mathbb{P}((\hat{P},V)(x)) < \epsilon(|x|)$.*

The condition of completeness says that if both the prover and verifier are good, then the verifier is convinced with a high probability that $x \in L$. Whenever $x \notin L$, the condition of soundness ensures that a good verifier, even in the presence of a dishonest prover, will not be convinced that $x \in L$.

A zero-knowledge proof is an interactive proof with the added condition that the interaction is zero-knowledge. The concept of being zero-knowledge is informally defined as the verifier learning nothing that he did not already know before the interaction other than the validity of some statement. However, simply constructing an interactive proof and showing that it is designed such that it is not possible for the verifier to deduce the secret information is not a sufficient argument and thus, the formal notion of a simulator needs to be introduced. A simulator simulates the execution of the verifier's algorithm V and outputs what V outputs, so if $(P,V)(x)$ enters an accept state with a probability $\mu(x)$ where $\mu(\cdot)$ is some polynomial function on the input x, then a simulator ought to enter an accept state with a probability $\mu(x)$. Now we define how exactly it is that a simulator is able to simulate an execution of an interactive protocol given that each round of computation for V depends on its initial input and auxiliary information or some output from P. A simulator Sim has *rewind* capabilities if it is able to rewind an interactive protocol to a previous step within a round or to a round within an execution. Let Sim_V be the simulation of V with a simulator that has rewind capability. Thus, a simulator enters an interactive proof with V and simulates a dummy prover that interacts with V, however, when needed the simulator can rewind the interaction to change its output or computation as the dummy prover.

The simulator Sim_V either outputs the same output that V outputs within the simulation or it outputs the fail symbol $\perp$ if some exceptional condition is met.[2] The motivation for the simulator is to show that the probability distribution with which Sim_V accepts is indistinguishable from the probability distribution that V would accept given a real-life interaction with a prover. Therefore, since the simulated interaction was done with a dummy prover (one that did not actually know the proof to the statement in question), then the verifier does not accomplish anything with the interactive proof that it wasn't already able to accomplish on its own [Lindell, 2017].

Let $\mathbb{P}_{Sim_V}(x)$ be the probability that a simulation of V with a dummy prover Sim enters an accept state on input x. A proof is zero-knowledge if for all probabilistic polynomial-time (PPT) algorithms V, there exists a PPT algorithm Sim_V such that the probability that $Sim_V(x)$ enters an accepting halt state is indistinguishable from the probability that $(P,V)(x)$ enters an accepting halt state. For any two algorithms G and G', two probabilities $\mathbb{P}_G$ and $\mathbb{P}_{G'}$ are are polynomially-indistinguishable if $|\mathbb{P}_G(x) - \mathbb{P}_{G'(x)}| \leq \epsilon(|x|)$ for a sufficiently large x [Goldwasser and Micali, 1982, Goldreich et al., 1987].

[2]Typically this refers to run-time, so if Sim_V exponentially exceeds the run-time of V, then the simulation is no longer accurate since we assume all V run in time polynomial bounded by the length of the input.

Definition 24 (Zero-Knowledge Interactive Proof)**.** *Let (P, V) be an interactive proof system and let L be a language over $\{0,1\}^*$. Let x be of sufficiently large length. (P, V) is said to be a zero-knowledge interactive proof for L if the following conditions are satisfied:*

- ***Completeness:*** *If $x \in L$, then $\mathbb{P}((P, V)(x)) \geq 1 - \epsilon(|x|)$,*
- ***Soundness:*** *For any protocol $\hat{P}$, if $x \notin L$, then $\mathbb{P}((\hat{P}, V)(x)) < \epsilon(|x|)$,*
- ***Zero-Knowledge:*** $|\mathbb{P}_{Sim_V}(x) - \mathbb{P}((P, V)(x))| \leq \epsilon(|x|)$

The completeness and soundness conditions are the same as in Definition 23. The condition of zero-knowledge states that the probabilities assigned to $Sim_V(x)$ and $(P, V)(x)$ are polynomially-indistinguishable. The above definition claims that the difference between the probabilities of each algorithm accepting is negligible, but this only describes *statistical* zero-knowledge. The concept of *perfect* zero-knowledge is satisfied when the probabilities associated with $Sim_V(x)$ and $(P, V)(x)$ are identical.

A proof that $x \in L$ can be called a *witness* y such that $T_L(x, y) = 1$ where T_L is a polynomial computable Boolean predicate associated with the language L such that $T_L(x, v) = 0$ for all v if $x \notin L$. y must have a length $|y| = \rho(|x|)$ for some polynomial function ρ, but does not need to be computable from x in polynomial time. We can refer to the set of all witnesses for the fact that $x \in L$ by $T_L(x) = \{y \,|\, T_L(x, y) = 1\}$. Next, we look at an example of a zero-knowledge proof as well as viewing the simulator algorithm which satisfies the condition of *zero-knowledge*.

4.2.1 Zero-Knowledge Protocol Example: Graph 3-Coloring

A simple graph $G(\mathbb{V}, E)$ consists of a nonempty, finite set of vertices $\mathbb{V}$ and a finite set E consisting of unordered pairs of distinct elements of $\mathbb{V}$ called edges. A graph is simple if there is at most one edge joining a given pair of vertices. A walk in a graph $G(\mathbb{V}, E)$ is a finite sequence of edges in which any two consecutive edges are adjacent. Furthermore, if all edges and vertices in a walk are distinct, then we refer to it is a path. A graph is connected iff there is a path between each pair of vertices [Wilson, 2010]. When refering to the graph 3-coloring problem, we restrict ourselves to the graphs defined as simple and connected.

Let Γ be a finite set, then $Sym(\Gamma)$ is the symmetric group of Γ, that is $Sym(\Gamma)$ is the group of permutations over the set Γ [Sagan, 2001]. Let $\pi, \tau \in Sym(\Gamma)$ then $\pi \circ \tau$ denotes the composition of the permutations π and τ such that $\pi \circ \tau(x) = \pi(\tau(x))$. Of note, if $|\Gamma| = z$, then $|Sym(\Gamma)| = z!$, so in the case of $\{1, 2, 3\}$, $Sym(\{1, 2, 3\}) = \begin{pmatrix} 1\,2\,3 & 1\,3\,2 \\ 2\,1\,3 & 2\,3\,1 \\ 3\,1\,2 & 3\,2\,1 \end{pmatrix}$.

For our protocols to function properly, a method of encryption known as a commitment scheme must be introduced. A commitment scheme allows a user to commit to a chosen value while keeping it hidden to others with the ability that they may reveal the value at a later time. This is typically done by a mathematical operation being performed on the value that is computationally infeasible to invert. We can define a commitment scheme as a one-to-one function F, and we let $c = F(x)$ denote the commitment of x. We let $D(c)$ denote the decommitment scheme of the value c, so if $c = F(x)$, then $D(c) = x$. We can refer to the act of sending $D(c)$ for a known commitment c as *opening* the commitment c. We also assume that the commitment scheme F produces $c_1 = F(x_1)$ and $c_2 = F(x_2)$, for $x_1 \neq x_2$, such that c_1 and c_2 are indistinguishable. That is one cannot deduce any information about x from a commitment $F(x)$. With the commitment scheme in place we can now provide an overview of a problem in **NP** and provide the interactive proof for a statement $x \in L$ where $L \in \mathbf{NP}$.

The language L is composed only of graphs whose vertices are colorable with three colors in a manner such that no two vertices joined by an edge have the same color. A graph is considered 3-colorable iff the graph is in L. Given a graph g, it is within the complexity class **NP** to determine whether $g \in L$ [Goldreich et al., 1991]. The witness y such that $T_L(g, y) = 1$ is a 3-coloring that satisfies the aforementioned requirements. Suppose someone knows such a coloring for a graph and wanted to prove to someone else that $g \in L$ without revealing any information about y. To do so, they could implement a zero-knowledge protocol.

Informally, a zero-knowledge protocol for proving that a graph is 3-colorable is an interactive protocol between a prover and a verifier. Both the prover and the verifier know the shape of the graph, that is they know the set of vertices as well as the set of edges. This is considered the common input. Only the prover knows a coloring for the graph, and the first step is for the prover to randomly pick a permutation over the set of three available colors and apply this permutation to the coloring that she knows for the graph. The prover then implements the commitment scheme to each vertex with a permuted coloring and sends the list of commitments to the verifier. The verifier randomly selects an edge of the graph and sends it to the prover. The prover responds by opening the commitments of the vertices in the edge selected by the verifier and the verifier checks to see if the colors of the vertices are not identical. If the prover knows a valid witness (a 3-coloring), then this process will always result with the verifier accepting, since the colors will never be identical. If the prover does not know a coloring for the graph, then she can never commit to a valid coloring of the graph, and thus there is at least one edge such that the colors of the vertices are identical. Therefore, if the prover does not know a coloring for the graph and is trying to deceive the verifier, then the verifier has a probability of $\frac{1}{m}$ to select the edge with identical vertices, where m is the total number of edges in the graph. In this case, the verifier will accept with a probability less than or equal to $1 - \frac{1}{m}$. By repeating the whole protocol, as described, $n \times m$ times, then the probability that the verifier accepts drops to less than $(1 - \frac{1}{m})^{nm}$ where n is the number of vertices in the graph. We now give a formal definition of the zero-knowledge protocol. For a finite set Γ, $x \in_R \Gamma$ says that x is distributed uniformly over the set Γ.

Protocol 1 (Zero-Knowledge Proof for 3-Coloring)**.** *Formally, the algorithm (P, V) can be described as follows:*

- *Common Input: Graph $G(\mathbb{V}, E)$ with $\mathbb{V} = \{v_1, ..., v_n\}$*
- *Prover's Input: A coloring $f : \mathbb{V} \to \{1, 2, 3\}$ such that for every $(v_i, v_j) \in E$ it holds that $f(v_i) \neq f(v_j)$.*
- *Protocol: Repeat $n \times |E|$ times*
  1. *P randomly selects a permutation $\pi \in_R Sym(\{1, 2, 3\})$ that is defined by $h(v_i) = \pi(f(v_i))$ for all $v_i \in \mathbb{V}$. P computes $c_i = F(h(v_i))$ for all $i \in 1, ..., n$. P sends the list $(c_1, ..., c_n)$ to V.*
  2. *V randomly chooses an edge $e \in_R E$ and sends e to P.*
  3. *Let $e = (v_i, v_j)$, then P sends $D(c_i)$ and $D(c_j)$ to V.*
  4. *Let $h(v_i) = D(c_i)$ and $h(v_j) = D(c_j)$. V checks whether $h(v_i), h(v_j) \in \{1, 2, 3\}$ and whether $h(v_i) \neq h(v_j)$, if not then V halts and outputs 0.*
- *If Step 4 does not halt in any iteration, then V outputs 1.*

Protocol 1 satisfies completeness, since if P has a coloring f, then V will accept every iteration with a probability of 1. The protocol satisfies soundness since $\mathbb{P}((P, V)(\overline{x})) < \epsilon(|\overline{x}|)$

where $\overline{x}$ represents a coloring that does not satisfy 3-colorability. To verify that Protocol 1 satisfies zero-knowledge, we need to construct a simulator Sim capable of simulating V for the above protocol. Recall, Sim takes as input V and the common input of Protocol 1 and runs its own algorithm with V operating as a subroutine.

Protocol 2 (Simulator for Protocol 1)**.** *$Sim_V(x)$ is defined as follows:*

1. *Repeat $n \times |E|$ times*
   a) *Sim sets $j = 1$ as an index.*
   b) *Sim randomly chooses an edge $(v_k, v_l) \in_R E$. Sim chooses $h(v_k) \in_R \{1, 2, 3\}$ and $h(v_l) \in_R \{1, 2, 3\}/\{h(v_k)\}$. For all $v_i \in V/\{v_k, v_l\}$ Sim sets the coloring of v_i to the null color 0.*
   c) *For every $i \in 1, ..., n$, Sim computes the commitments $c_i = F(h(v_i))$*
   d) *Sim sends $(c_1, ..., c_n)$ to V. Let $e \in E$ be the reply from V.*
   e) *If $e = (v_k, v_l)$ then Sim sends V $D(c_k)$ and $D(c_l)$ and completes this iteration.*
   f) *If $e \neq (v_k, v_l)$, then Sim sets $j := j + 1$. If $j = n \times |E|$, then Sim outputs $\perp$; else, Sim returns to Step 2b).*
2. *Sim outputs whatever V outputs on the final iteration.*

In Protocol 2, the act of moving from Step 2f) to 2b) is the simulator rewinding the interaction. As can be seen in the protocol, if the edge that the simulator generated a coloring for is not the edge chosen by V, then Sim rewinds the interaction to the point where Sim chooses an edge and proceeds from there. Clearly, Sim runs in polynomial time since there are at most $n \times |E|$ iterations that each run at most $n \times |E|$ times. To prove that the probability assigned to Sim entering an accept state, $\mathbb{P}(Sim_V(x))$, is indistinguishable from $\mathbb{P}((P, V)(x))$ we need to prove that the probability that Sim_V outputs $\perp$ is negligible.

Given that all commitments are indistinguishable, the probability that a single run of an iteration succeeds is $\frac{1}{|E|}$. So for one iteration, Sim outputs $\perp$ with a probability of $(1 - \frac{1}{|E|})^{n \cdot |E|}$, which is less than $\exp(-n)$. For the total $n \times |E|$ iterations in Protocol 2, the probability that Sim outputs $\perp$ is less than $n \times |E| \times e^{-n}$, which is negligible [Lindell, 2017]. Recall, Sim accepts and rejects depending on whether the subroutine of V accepts or rejects within Sim, so it holds that $|\mathbb{P}(Sim_V(x)) - \mathbb{P}((P, V)(x))| < \epsilon(|x|)$.

The computational concepts described above will need to be represented in the formal language of a logic capable of modeling zero-knowledge protocols. An issue of concern for the logic would be the representation of probabilities in an epistemic logic. Previous work regarding probability and epistemic logic has been done by van Eijck and Schwarzentruber [2014] and Kooi et al. [2009]. We follow van Eijck and Schwarzentruber in turning the epistemic cryptography models of Chapter 3 into epistemic probability models where we can equate knowledge with a high level of certainty, thus allowing us to model zero-knowledge proofs where a verifier is convinced with a high level of certainty and where propositions are indistinguishable based on their probability distributions.

4.3 Cryptographic Probability Logic

The logic established in this section will aim to accomplish similar protocol modeling techniques as previously described while also implementing a syntax robust enough to accommodate the intricate conditions of a zero-knowledge protocol. We will implement a parsing function so we

can utilize the identity relation amongst cryptographic messages. We will also refine the basic propositions of the logic to allow us to reason about more complex interactive protocols. There is something to be said about the notion of knowledge utilized in DEL compared to the type of knowledge necessary in an interactive proof system.

4.3.1 Epistemic Probability Models

The objective of converting epistemic models into probability models assigns each agent to an equivalence relation on the set of worlds as well as a function from worlds to the set of positive rationals $\mathbb{Q}^+$. Therefore, we can link an agent's epistemic access to information via subjective probabilities. Other logics designed to account for probabilistic reasoning are subtly different in the fact that rather than express statements about the probability an agent assigns to a given world where propositions may or may not hold, they contain terms that represent an agent's expectation of the truth of a proposition at a given world [French et al., 2019a,b]. This section will first introduce the concepts of lotteries which assign rational numbers to each world in a model, as well defining an epistemic probability model before converting the epistemic cryptography models of Chapter 3 into epistemic probability models.

In the traditional approach of epistemic probability logic [van Eijck and Schwarzentruber, 2014], knowledge is equated with absolute certainty. Therefore, to say an agent i knows a proposition φ is to say that the probability that i assigns to φ equals 1. We will adjust this notion later so that i knows a proposition φ iff the probability i assigns to φ equals α for $\alpha \in (1-\epsilon(n), 1]$ for a sufficiently large n. That is, the probability i assigns to φ is greater than $1-\epsilon(n)$.

We define a probability model $\mathcal{M}$ for a set $\mathbf{P}$ of propositions and agents A as a tuple $(W, R, \mathcal{V}, \mathbb{L})$, where $(W, R, \mathcal{V})$ is a standard Kripke model. $\mathbb{L} : A \to (W \to \mathbb{Q}^+)$ is a function which assigns to each agent, a lottery representing the subjective probabilities that the agent assigns to worlds.

Definition 25 (Lotteries)**.** *A W-lottery $\mathbb{L}$ is a function from W to the set of positive rationals $\mathbb{Q}^+$. A W-lottery is bounded on $U \subseteq W$ if $\sum_{u \in U} \mathbb{L}(u) < \infty$.*

Definition 26 (Epistemic Probability Model)**.** *An epistemic probability model $\mathcal{M}$ is a tuple $(W, R, \mathcal{V}, \mathbb{L})$ where $W, R, \mathcal{V}$ are defined as a standard Kripke model and $\mathbb{L}$ is a function that assigns to every agent $i \in A$ a W-lottery that is bounded on every R_i equivalence class.*

An epistemic probability model is normalized if $\mathbb{L}_i$ restricted to ξ is a probability measure, as defined in Definition 11, for all agents $i \in A$ and for all R_i equivalence classes ξ. Since $\mathbb{L}_i$ is bounded on R_i for all $i \in A$, then all epistemic probability models can be normalized.

We can now introduce the language of epistemic probability logic $\mathcal{L}_{EPL}$ for a set of propositions $\mathbf{P}$ and a set of agents A.

Definition 27 (Syntax of EPL)**.** *Let p range over $\mathbf{P}$, i over A, and q over $\mathbb{Q}$.*

$$\begin{array}{lcl} \varphi & := & \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid t_i \geq 0 \mid t_i = 0 \\ t_i & := & q \mid q \cdot \mathbb{P}_i\varphi \mid t_i + t_i \end{array}$$

Where $t_i + t_i$ assumes that both agent indices are identical.

The term t_i generates linear expressions dealing with the subjective probabilities of i, so we say $t_i = 0$ or $t_i > 0$ is an i-probability formula. The expression $\mathbb{P}_i(\varphi) = q$ expresses that the probability of φ according to i is q. Besides the usual abbreviations for disjunction,

material implication, and equivalence statements, we can define useful abbreviations relating to probabilities using the syntax of the language as follows:

$$\begin{array}{rcl} t \geq t' & := & t + (-1)t' \geq 0 \\ t < t' & := & \neg t \geq t' \\ t > t' & := & \neg t' \geq t \\ t \leq t' & := & t' \geq t \\ t \neq t' & := & \neg t = t' \\ \mathbb{P}_i(\varphi|\varphi') = q & := & \mathbb{P}_i(\varphi') > 0 \wedge q \cdot \mathbb{P}_i(\varphi') = \mathbb{P}_i(\varphi \wedge \varphi') \end{array}$$

$\mathbb{P}_i(\varphi|\varphi') = q$ is an expression stating that according to i, the probability of φ, conditional on φ' is q.

Definition 28 (Semantics of Epistemic Probability Logic)**.** *Given an epistemic probability model $\mathcal{M} = (W, R, \mathcal{V}, \mathbb{L})$ and a world $w \in W$:*

$$\begin{array}{lcl} \mathcal{M}, w \vDash \top & \Leftrightarrow & \textit{always} \\ \mathcal{M}, w \vDash p & \Leftrightarrow & p \in \mathcal{V}(w) \\ \mathcal{M}, w \vDash \neg\varphi & \Leftrightarrow & \mathcal{M}, w \nvDash \varphi \\ \mathcal{M}, w \vDash \varphi \wedge \varphi' & \Leftrightarrow & \mathcal{M}, w \vDash \varphi \textit{ and } \mathcal{M}, w \vDash \varphi' \\ \mathcal{M}, w \vDash t_i \geq 0 & \Leftrightarrow & [\![t_i]\!]^{\mathcal{M}}_w \geq 0 \\ \mathcal{M}, w \vDash t_i = 0 & \Leftrightarrow & [\![t_i]\!]^{\mathcal{M}}_w = 0 \\ [\![q]\!]^{\mathcal{M}}_w & := & q \\ [\![q \cdot \mathbb{P}_i\varphi]\!]^{\mathcal{M}}_w & := & q \times \mathbb{P}^{\mathcal{M}}_{i,w}(\varphi) \\ [\![t_i + t'_i]\!]^{\mathcal{M}}_w & := & [\![t_i]\!]^{\mathcal{M}}_w + [\![t'_i]\!]^{\mathcal{M}}_w \end{array}$$

where

$$\mathbb{P}^{\mathcal{M}}_{i,w}(\varphi) = \frac{\sum\{\mathbb{L}_i(v) \mid \langle w, v\rangle \in R_i \textit{ and } \mathcal{M}, v \vDash \varphi\}}{\sum\{\mathbb{L}_i(v) \mid \langle w, v\rangle \in R_i\}}$$

The definition of $\mathbb{P}^{\mathcal{M}}_{i,w}$ is well-defined since $\mathbb{L}_i(v) > 0$ for all $v \in W$ and by Definition 26, all lotteries are bounded over the equivalence relation of the agent, so $\sum\{\mathbb{L}_i(v) \mid \langle w, v\rangle \in R_i\} < \infty$.

As mentioned, when integrating probabilities into the cryptographic models, our definition of knowledge will not require absolute certainty, only a *high level* of certainty. However, for present purposes of explanation we continue with the notion that $K_i\varphi := \mathbb{P}_i\varphi = 1$. Consider a model with two agents a and b and two propositions p and r as seen in Figure 4.1. This model has the lotteries of $\mathbb{L}_a = \{\mathbf{0} : \frac{1}{8}, \mathbf{1} : \frac{1}{4}, \mathbf{2} : \frac{1}{2}, \mathbf{3} : \frac{1}{8}\}$ and $\mathbb{L}_b = \{\mathbf{0} : \frac{1}{3}, \mathbf{1} : \frac{1}{3}, \mathbf{2} : \frac{1}{6}, \mathbf{3} : \frac{1}{6}\}$ and the accessibility relations of a and b denoted by the solid and dashed lines, respectively. The lines for reflexive relations are omitted, but we assume they hold for all agents at all worlds.

Our example model shows a situation in which at world $\mathbf{1}$, $K_b r$ holds. At all worlds accessible from $\mathbf{1}$ by b, r is true, and thus the probability that b assigns to r at world $\mathbf{1}$ is 1. Likewise, $K_b p$ is false at world $\mathbf{1}$ since $\frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2} < 1$.

Proposition 8 (van Eijck and Schwarzentruber [2014])**.** *Let φ be a formula of standard epistemic logic. The formula φ is satisfiable in a standard epistemic model iff $tr(\varphi)$ is satisfiable in an epistemic probability model. tr is defined by $tr(K_i\varphi) = \mathbb{P}_i tr(\varphi) = 1$*

Proof. $\Rightarrow$) Suppose φ is satisfiable in a standard epistemic model, then by the finite model property there is a finite standard epistemic model for φ [Blackburn et al., 2001]. This model

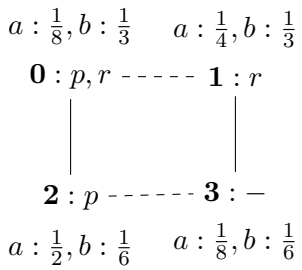


Figure 4.1: Example of a model $\mathcal{M}$ in Epistemic Probability Logic

can be transformed into an epistemic probability model $\mathcal{M} = (W, R, \mathcal{V}, \mathbb{L})$ by adding fake lotteries that assign to each agent a W-lottery of 1 to every world. As the model is finite, it is guaranteed that the W-lottery $\mathbb{L}_i$ is bounded on every R_i equivalence class for all $i \in A$. We can then prove that $tr(\varphi)$ is true in $\mathcal{M}$.

$\Leftarrow$) Extract a standard epistemic model from a epistemic probability model by removing all lotteries and proving that φ is satisfiable. □

We can now define epistemic probability action models, which will allow us to update the epistemic probability models. With update models in place, we can model dynamic scenarios in multi-agent settings with subjective probabilities.

An epistemic probability action model $\mathfrak{E}$ is defined as $(\mathcal{E}, \sim, \mathsf{Pre}, \mathsf{Pos}, \mathbb{L})$ where $\mathcal{E}$ is a set of actions, $\sim$ is the function from agents to pairs of actions serving as the accessibility relation, Pre and Pos are the preconditions and postconditions of the model that assign propositions in $\mathbf{P}$ to formulas in $\mathcal{L}_{EPL}$. Finally, $\mathbb{L}$ is a function which assigns to each agent $i \in A$ an $\mathcal{E}$-lottery that is bounded on every $\sim_i$ equivalence class.

Definition 29 (Epistemic Probability Action Models)**.** *Let $\mathfrak{E}$ be an epistemic probability action model defined by the tuple $(\mathcal{E}, \sim, \mathsf{Pre}, \mathsf{Pos}, \mathbb{L})$ where $\mathcal{E}$ is a finite set of actions. $\sim$ is a function from agents in A to pairs of actions in $\mathcal{E}$ such that $e_1 \sim_i e_2$ denotes i's indistinguishability of the actions e_1 and e_2. Pre is a function which assigns to each action $e \in \mathcal{E}$ a formula $\varphi \in \mathcal{L}_{EPL}$ such that φ holds in all worlds that e can be executed. $\mathsf{Pos} : \mathcal{E} \to \mathsf{SUB}(\mathcal{L}_{EPL})$ maps actions in $e \in \mathcal{E}$ to a substitution from $\mathbf{P}$ to formulas in $\mathcal{L}_{EPL}$. The lottery function is defined as $\mathbb{L} : A \to (\mathcal{E} \to \mathbb{Q}^+)$ which assigns an $\mathcal{E}$-lottery to each agent $i \in A$ such that for all actions e in a $\sim_i$ equivalence class E: $\sum_{e \in E} \mathbb{L}_i(e) < \infty$.*

The substitution function in the postcondition Pos binds propositions to formulas so we can say that propositions are bound to $\top$ or $\bot$ to determine their truth value. For example, a postcondition $p := \top$ binds p to true, so the result of the action executed makes p true. We can now define an update model for epistemic probability models.

Definition 30 (Probability Update Model)**.** *Let i range over the set of agents A and p over $\mathbf{P}$. Let $\mathfrak{E}$ be an epistemic probability action model and $\mathcal{M}$ be an epistemic probability model. We denote the update model as $\mathcal{M} \circ \mathfrak{E} = (W', R', \mathcal{V}', \mathbb{L}')$ defined as follows:*

- $W' := \{\langle w, e\rangle \mid \mathcal{M}, w \vDash \mathsf{Pre}(e)\}$
- $R_i' := \{(\langle w, e\rangle, \langle u, e'\rangle \mid \langle w, v\rangle \in R_i \text{ and } e \sim_i e'\}$
- $\mathcal{V}'(w) := \{p \mid \mathcal{M}, w \vDash \mathsf{Pos}(e)(p)\}$
- $\mathbb{L}_i'(\langle w, e\rangle) = \mathbb{L}_i(w) \times \mathbb{L}_i(e)$

If the initial model $\mathcal{M}$ and the action model $\mathfrak{E}$ are both normalized, then $\mathcal{M} \circ \mathfrak{E}$ is an epistemic probability model.

We can now define a dynamic epistemic probability logic (DEPL) by adding actions to the syntax of our language $\mathcal{L}_{EPL}$ as well as the corresponding semantic clause. We will append the standard formula $[\mathfrak{E}, e]\varphi$ to the syntax of $\mathcal{L}_{EPL}$. We can now refer to the logic as DEPL.

Definition 31 (Syntax of $\mathcal{L}_{DEPL}$)**.**

$$\begin{array}{lcl} \varphi & := & \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid t_i \geq 0 \mid t_i = 0 \mid [\mathfrak{E}, e]\varphi \\ t_i & := & q \mid q \cdot \mathbb{P}_i\varphi \mid t_i + t_i \end{array}$$

Definition 32 (Semantics for $\mathcal{L}_{EPL}$)**.** *We define the semantics for each formula in the language as in Definition 28 with the addition of the following:*

$$\mathcal{M}, w \vDash [\mathfrak{E}, e]\varphi \;\Leftrightarrow\; \textit{if } \mathcal{M}, w \vDash \mathsf{Pre}(e) \textit{ then } \mathcal{M} \circ \mathfrak{E}, \langle w, e\rangle \vDash \varphi$$

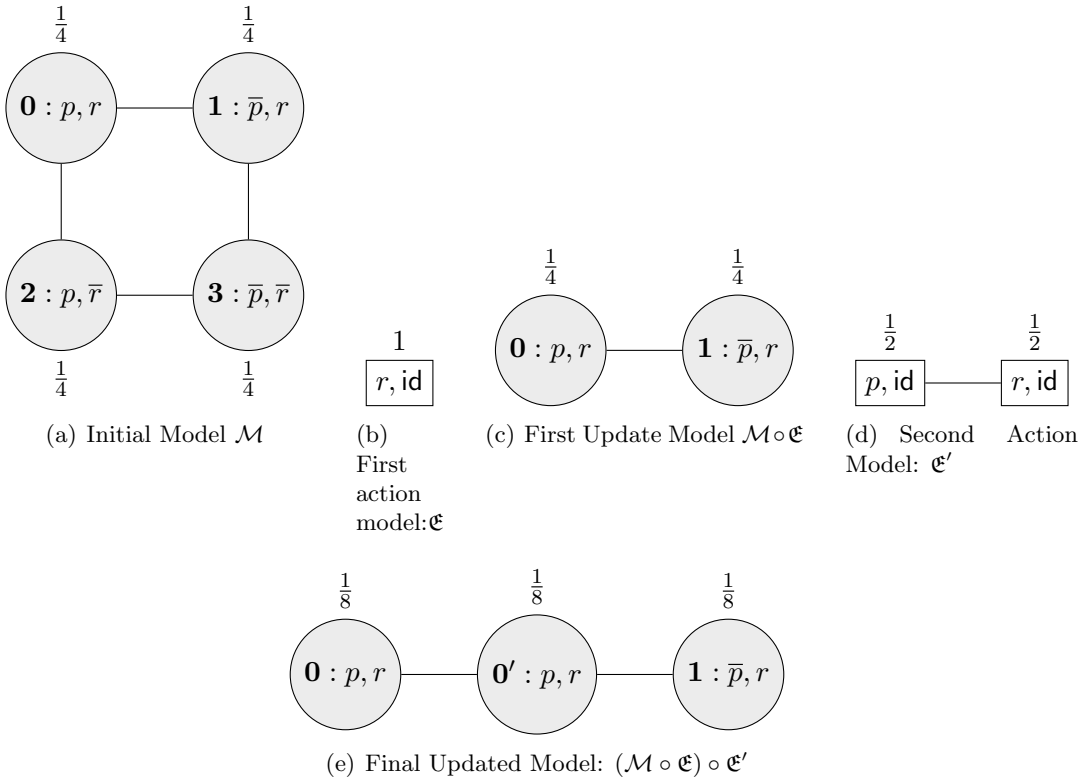
4.3.1.1 Probability Puzzle

We can consider an example based in probability theory to see the effect of dynamic models in probabilistic reasoning amongst agents. The following probability puzzle originates in Carroll [1895]. Suppose Alice has an opaque urn with two compartments and a marble in each compartment such that she cannot see either marble. The marbles can either be black or white, and Alice knows this. Alice pulls a marble out of one of the compartments of the urn and inspects it, to see that it is white, she then places the marble back into the urn. Lastly a white marble is removed by a third-party and shown to Alice. What probability does Alice assign to the marble remaining in the urn also being white?[3] We can model Alice's epistemic uncertainty at each stage of the puzzle as well as update our model with actions as described.

Let p and r refer to the two marbles in the urn. Since Alice is uncertain to the color of the marbles we denote p for white and $\neg p$ for black, likewise for the second marble and the proposition r. Furthermore, we will attach the probabilities with which Alice considers each world possible. There is an illustration of the initial model in Figure 4.2(a). Let us note that in the initial model Alice assigns the proposition that each marble in the urn is white a probability of $\frac{1}{2}$. Each world has a lottery of $\frac{1}{4}$, but when each proposition p and r is considered individually, the probability that Alice assigns to p and r is $\frac{1}{2}$. Without loss of generality, let r be the marble in the compartment that Alice observes. The action models are illustrated such that the precondition is listed before the postcondition. Figure 4.2(b) lists r as the precondition which means that the action takes place in only those worlds where r holds, Figure 4.2(b) also lists a probability of 1, representing the certainty of Alice observing the white marble in the compartment of the urn.

By updating the model in Figure 4.2(a) with the action model depicted in Figure 4.2(b), we obtain the resulting model of Figure 4.2(c) by applying the action in all worlds where the precondition holds and by multiplying the lotteries of the initial worlds by the lottery of the action. The action of removing a white marble is represented by the action model in Figure 4.2(d) where Alice cannot distinguish between removing the marble p or the marble r by the third-party. The precondition for removing p and seeing that it is white is p, and likewise the precondition for removing r and observing that it is white is r. This means the action of removing p and observing that it is white can only be executed in world **0**, while the action of removing r and observing that it is white can be executed in both **0** and **1**. Updating the

[3]This is a single agent version of the puzzle which appears in van Eijck and Schwarzentruber [2014] making it more in line with the original puzzle posed by Carroll [1895].



(a) Initial Model $\mathcal{M}$ (b) First action model: $\mathfrak{E}$ (c) First Update Model $\mathcal{M} \circ \mathfrak{E}$ (d) Second Action Model: $\mathfrak{E}'$

(e) Final Updated Model: $(\mathcal{M} \circ \mathfrak{E}) \circ \mathfrak{E}'$

Figure 4.2: Models of the Puzzle updates

model in Figure 4.2(c) with the action model $\mathfrak{E}'$ results in the model of Figure 4.2(e) with the standard changes to the probabilities and propositional values at each world.

In the final updated model, at all worlds Alice assigns to the proposition that the marble still in the urn is white the probability $\frac{2}{3}$ since all worlds $\mathbf{0}, \mathbf{0}'$, and $\mathbf{1}$ are accessible by Alice and $\frac{\sum\{\mathbb{L}_a(\mathbf{0}), \mathbb{L}_a(\mathbf{0}')\}}{\sum\{\mathbb{L}_a(\mathbf{0}), \mathbb{L}_a(\mathbf{0}'), \mathbb{L}_a(\mathbf{1})\}} = \frac{\frac{1}{4}}{\frac{3}{8}} = \frac{2}{3}$. So, the act of observing a white marble in the urn and then removing a white marble from the urn causes Alice to increase her certainty that the remaining marble is white from $\frac{1}{2}$ in the initial model to $\frac{2}{3}$ in the final updated model.[4]

4.3.1.2 Applying Probability Models to Cryptographic Models

Now that we have introduced the concept of dynamic epistemic probability models and we are comfortable performing probabilistic reasoning within the models, we can apply lotteries and probability action models to our cryptographic logic thus allowing us to model more complex cryptographic scenarios.

[4]To verify that this result is correct, take the conditional probability $\mathbb{P}[D_2 = \mathrm{w} \mid D_1 = \mathrm{w}] = \frac{\mathbb{P}[D_2 = \mathrm{w} \cap D_1 = \mathrm{w}]}{\mathbb{P}[D_1 = w]}$ which is the probability that the second marble drawn from the urn is white conditional on the fact that the first marble drawn is white. The probability that both marbles were white is simply the probability that the unobserved marble was white, since the probability that the observed marble is white is 1 we have $\frac{1}{2} \times 1 = \frac{1}{2}$. The probability that the marble removed was white is the sum of the probabilities of removing a white marble from both initial cases: the unobserved marble is white and the unobserved marble is black. As stated, in the first case the probability is $\frac{1}{2}$, and in the second case the probability is $\frac{1}{2} \times \frac{1}{2}$, which are summed to $\frac{3}{4}$. Thus, $\mathbb{P}[D_2 = \mathrm{w} \mid D_1 = \mathrm{w}] = \frac{2}{3}$.

Let us start by defining a cryptographic probability model as the tuple $(W, R, I, AL, \mathbb{L})$ where W, R, I, AL are defined as in Definition 16 and $\mathbb{L}$ is a function which assigns to each agent $i \in A$ a W-lottery bounded on the set of equivalence classes in R_i. Likewise, we can modify the action models in the same manner by defining a cryptographic probability action model $\mathfrak{A}$ as $(\mathcal{A}, \sim, \mathsf{Pre}, \mathsf{Pos}_I, \mathsf{Pos}_{AL}, \mathbb{L})$ where $\mathcal{A}, \sim, \mathsf{Pre}, \mathsf{Pos}_I, \mathsf{Pos}_{AL}$ are defined as before in Definition 19 and $\mathbb{L}$ assigns to each agent $i \in A$ an $\mathcal{A}$-lottery bounded on the set of equivalence classes in $\sim_i$.

For a cryptographic probability model $\mathcal{M}$ and a cryptographic probability action model $\mathfrak{A}$, we can denote the product of update $\mathcal{M}$ with $\mathfrak{A}$ as $\mathcal{M} \circ \mathfrak{A} = (W', R', I', AL', \mathbb{L}')$. Letting i range over the agents in A, w over the worlds in W, and σ over the actions in $\mathcal{A}$; W', R', I', AL' can all be defined as in Definition 20 with the addition that $\mathbb{L}'_i(\langle w, \sigma \rangle) = \mathbb{L}_i(w) \times \mathbb{L}_i(\sigma)$. In the next section we will define the syntax and semantics of cryptographic probability logic and define knowledge in terms of the new probability semantics.

4.3.2 Syntax and Semantics

Much of the syntax for cryptographic probability logic (CPL) will remain the same from Chapter 3 with the addition of linear expressions in the language $\mathcal{L}_{CPL}$. The set of messages M is still a collection of numerical values and keys that can be composed into encrypted forms or concatenated forms.

Definition 33 (Messages)**.** *Let M be a finite set of messages. A message $m \in M$ is defined as:*

$$m ::= \; n \mid k \mid k^+ \mid k^- \mid \; \{m\}_k \mid (m, m')$$

where the keys have appropriate subscripts when necessary.

Definition 34 (Propositions)**.** *The set Φ^A contains the basic set of propositions, defined as:*

$$p ::= \; \mathsf{has}_i m \mid \mathsf{const}_i m \mid mk(\sigma) \mid m =_i m'$$

The propositions remain the same with the addition of an identity proposition indexed for each agent $i \in A$. The new proposition of identity should be interpreted as 'according to i, m and m' are identical'.

Definition 35 (Syntax)**.** *Given agents $i \in A$, propositions $p \in \Phi^A$, actions $\sigma \in \mathcal{A}$, and rationals $q \in \mathbb{Q}$; formulas of $\mathcal{L}$ are defined by:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\mathfrak{A}, \sigma]\varphi \mid [\mathfrak{A}]^*\varphi \mid t_i \geq 0 \mid t_i = 0$$

$$t_i ::= q \mid q \cdot \mathbb{P}_i\varphi \mid t_i + t_i$$

Standard abbreviations for disjunction, material implication, equivalence, and inequality expressions are assumed as well as probability expressions for conditional probability. The indices in $t_i + t_i$ are assumed to be identical.

For simplicity we can design a set of construction rules that contains both symmetric keys and asymmetric keys, thus allowing both types of encryption within a given model.

Definition 36 (Construction Rules $\mathbb{C}$)**.** *Let m range over the set of messages M and k range over the set of keys in M. We let k_i^+, k_i^- denote the public and private key of agent i, respectively.*

$$\frac{m}{m} \quad \frac{m \quad k}{m_k} \quad \frac{m_k \quad k}{m} \quad \frac{m \quad k_i^+}{m_{k_i^+}} \quad \frac{m \quad k_i^-}{m_{k_i^-}} \quad \frac{m_{k_i^+} \quad k_i^-}{m} \quad \frac{m_{k_i^-} \quad k_i^+}{m} \quad \frac{m \quad m'}{(m, m')} \quad \frac{(m, m')}{m} \quad \frac{(m, m')}{m'}$$

While the const proposition captures what messages an agent is capable of constructing by using the construction rules and their available information set, we need to establish a notion of identity within the logic. Cryptographic parsing will capture how messages are perceived by an agent given their information sets. When an encrypted message is received by an agent, they will be able to decide whether it is identical to another message they possess if they are capable of decrpyting the message, otherwise it is an abstract, un-readable message ■.

Definition 37 (Cryptographic parsing)**.** *Let m be a message. The cryptographic parsing function $\ulcorner\cdot\urcorner : M \times W \times A \to M \cup \{\blacksquare\}$ uses an agent's information set at a world to determine how a message can be interpreted by the agent. We define the value of the cryptographic parsing function $\ulcorner\cdot\urcorner_i^w$ by induction on the structure of a message m.*

$$
\begin{aligned}
\ulcorner n \urcorner_i^w &:= n \\
\ulcorner k \urcorner_i^w &:= k \\
\ulcorner \{m\}_k \urcorner_i^w &:= \begin{cases} \ulcorner m \urcorner_i^w & \text{if } k \in I_{w,i} \text{ or } k \in \overline{I_{w,i}} \\ \blacksquare & \text{otherwise,} \end{cases} \\
\ulcorner \{m\}_{k_j^-} \urcorner_i^w &:= \begin{cases} \ulcorner m \urcorner_i^w & \text{if } k_j^+ \in I_{w,i} \text{ or } k_j^+ \in \overline{I_{w,i}} \\ \blacksquare & \text{otherwise,} \end{cases} \\
\ulcorner \{m\}_{k_j^+} \urcorner_i^w &:= \begin{cases} \ulcorner m \urcorner_i^w & \text{if } k_j^- \in I_{w,i} \text{ or } k_j^- \in \overline{I_{w,i}} \\ \blacksquare & \text{otherwise,} \end{cases} \\
\ulcorner (m, m') \urcorner_i^w &:= (\ulcorner m \urcorner_i^w, \ulcorner m' \urcorner_i^w)
\end{aligned}
$$

If $\mathcal{M}, w \not\models \mathsf{has}_i k$ and $\mathcal{M}, w' \not\models \mathsf{has}_i k$, then $\ulcorner \{m\}_k \urcorner_i^w = \blacksquare = \ulcorner \{m'\}_k \urcorner_i^{w'}$. Two plaintexts that are encrypted with the same symmetric key k are parsed by $\ulcorner\cdot\urcorner$ to the same abstract message ■ when the agent does not know the decryption key. With the parsing function, we can now represent how messages are perceived by individual agents and thus we can establish an identity relation indexed to each agent corresponding to the parsing function.

Definition 38 (Semantics of basic propositions)**.** *Let $\mathcal{M}$ be a model for $\mathcal{L}_{CPL}$ as described in Section 4.3.1.2 and $w \in W$.*

$$\mathcal{M}, w \vDash \mathsf{has}_i m \;\Leftrightarrow\; m \in I_{w,i}$$

$$\mathcal{M}, w \vDash \mathsf{const}_i m \;\Leftrightarrow\; m \in \overline{I_{w,i}}$$

$$\mathcal{M}, w \vDash mk(\sigma) \;\Leftrightarrow\; \sigma \in AL(w)$$

$$\mathcal{M}, w \vDash m =_i m' \;\Leftrightarrow\; \ulcorner m \urcorner_i^w = \ulcorner m' \urcorner_i^w$$

The identity relation over the set of messages holds for an agent iff that agent's cryptographic parsing function assigns the two messages to the same value. With the semantics of the basic propositions defined we can define the semantics for formulas in the language $\mathcal{L}_{CPL}$.

Definition 39 (Semantics of CPL)**.** *Given a cryptographic probability model* $\mathcal{M} = (W, R, I, AL, \mathbb{L})$ *and* $w \in W$*:*

$$
\begin{array}{lcl}
\mathcal{M}, w \vDash \top & \Leftrightarrow & \textit{always} \\
\mathcal{M}, w \vDash \neg\varphi & \Leftrightarrow & \mathcal{M}, w \nvDash \varphi \\
\mathcal{M}, w \vDash \varphi \wedge \varphi' & \Leftrightarrow & \mathcal{M}, w \vDash \varphi \textit{ and } \mathcal{M}, w \vDash \varphi' \\
\mathcal{M}, w \vDash [\mathfrak{A}, \sigma]\varphi & \Leftrightarrow & \mathcal{M}, w \vDash \mathsf{Pre}(\sigma) \textit{ implies } \mathcal{M} \circ \mathfrak{A}, \langle w, \sigma \rangle \vDash \varphi \\
\mathcal{M}, w \vDash [\mathfrak{A}]^*\varphi & \Leftrightarrow & \textit{for all } n \in \mathbb{N} : \mathcal{M}, w \vDash [\mathfrak{A}]^n\varphi \\
\mathcal{M}, w \vDash t_i \geq 0 & \Leftrightarrow & [\![t_i]\!]^{\mathcal{M}}_w \geq 0 \\
\mathcal{M}, w \vDash t_i = 0 & \Leftrightarrow & [\![t_i]\!]^{\mathcal{M}}_w = 0 \\
[\![q]\!]^{\mathcal{M}}_w & := & q \\
[\![q \cdot \mathbb{P}_i\varphi]\!]^{\mathcal{M}}_w & := & q \times \mathbb{P}^{\mathcal{M}}_{i,w}(\varphi) \\
[\![t_i + t'_i]\!]^{\mathcal{M}}_w & := & [\![t_i]\!]^{\mathcal{M}}_w + [\![t'_i]\!]^{\mathcal{M}}_w \\
 & \textit{where} & \\
\mathbb{P}^{\mathcal{M}}_{i,w}(\varphi) & = & \dfrac{\sum\{\mathbb{L}_i(v) \mid \langle w, v \rangle \in R_i \textit{ and } \mathcal{M}, v \vDash \varphi\}}{\sum\{\mathbb{L}_i(v) \mid \langle w, v \rangle \in R_i\}}
\end{array}
$$

4.3.2.1 Knowledge in CPL

Given a CPL model, we need to define what it means for an agent within this model to know a proposition φ. The simplified[5] epistemic probability logic [van Eijck and Schwarzentruber, 2014] equates knowledge with absolute certainty, i.e., a probability of 1 assigned to φ. For our cryptographic purposes we won't need such a high threshold for knowledge, instead we can say that the knowledge operator $K_i\varphi$ is an abbreviation for $\mathbb{P}_i\varphi = \alpha$ where $\alpha \in (1 - \epsilon(n), 1]$ for a sufficiently large n.

van Ditmarsch and Kuijer [2019] analyze knowledge without absolute certainty within modal logic by enriching standard epistemic logic with a set of filters creating Epistemic Logic with Filters (ELF). The models of ELF rely on the use of *filters* to ensure that knowledge of a proposition φ is equivalent to φ holding in a *sufficiently large* number of possible worlds. The notion of sufficiently large relies on the possible worlds which are both considered by the agents in the model and the worlds which are contextually relevant for scenarios. CPL differs from ELF with respect to how many worlds are necessary by virtue of the lotteries assigned to worlds in models of CPL. As will be seen in later sections, we can arrive at knowledge in CPL even when only considering two possible worlds with diametrically opposed valuations for a given proposition so long as the lottery assigned to one of the worlds is negligible.

The traditional view of knowledge in epistemic logics and computer science is that of implicit knowledge which states that an agent i knows a proposition φ if φ is true in all states that are accessible by i from the current state. Using this traditional definition, there is a knowledge axiom which states $K_i\varphi \to \varphi$, interpreted as meaning 'i only knows true statements'. To reason about resource-limited distributed computations (PPT algorithms) and specifically interactive protocols, implicit knowledge will be inadequate for accurate modeling techniques [Fischer and

[5]While the epistemic probability logic introduced by [van Eijck and Schwarzentruber, 2014] is certainly the most simplified version in terms of semantics, it is not the only attempt made at conjoining probabilistic reasoning and epistemic logics. See: Baltag and Smets [2008], Fagin and Halpern [1994], Fagin et al. [1990], Kooi [2003], Kooi et al. [2009] for other variations of probability epistemic logic that aim to model different types of probabilistic reasoning. The choice to use the epistemic probability logic introduced by van Eijck and Schwarzentruber as the foundation of CPL was made to keep the mechanical focus of the logic on the cryptographic notions rather than elaborate semantics designed to allow for probabilistic reasoning. van Eijck and Schwarzentruber provided a method that could transform any epistemic model into a probabilistic model, so it requires the fewest changes to the underlying model structure, making it an ideal system to enrich the cryptographic variant of DEL.

Zuck, 1987]. Fischer and Zuck list a number of reasons why new conceptions of knowledge like probabilistic knowledge need to be introduced for modeling interactive protocols and we sketch a few of those reasons here.

The central notion behind most cryptographic protocols and computations is that they are computationally difficult or infeasible to accomplish in a reasonable amount of time. Despite this, there is potential for an agent to simply guess or accidentally determine some value for a computationally infeasible function. In a case such as an interactive proof, the prover's knowledge might not be that she knows how to compute discrete logarithms but rather that it happens sometimes for an unknown reason. We can refer to such a situation as accidental knowledge.

Furthermore in an interactive proof that requires probabilistic reasoning, it might be the case that the verifier sees a proposition φ at 99% of the worlds accessible to it, but in the remaining 1% of the worlds, φ is false. Under the definition of implicit knowledge, it would fail to hold that the verifier *knows* φ because φ is not always true in the worlds accessible by the verifier. Therefore, the notion of knowledge used in interactive proofs must forgo the knowledge axiom and use a definition of knowledge that is traditionally reserved for belief. This way, a notion of probabilistic knowledge is not absolute but corresponds to a degree of certainty.

The application of probabilistic knowledge applied to cryptographic protocols allows specification of these protocols with a high-level of abstraction in terms of knowledge and probability. Thus, reasoning about correctness of cryptographic protocols can then be done with axioms and inference rules for probabilistic knowledge [Halpern and Tuttle, 1989].

As a further consideration for knowledge in logics with probability, the lottery paradox demonstrates that basic notions of probabilistic reasoning lead to a contradiction [Kyburg Jr., 1961]. With our definition of knowledge in CPL, we can accept that an agent knows a proposition φ if the probability that the agent assigns to φ is greater than $1-\epsilon(n)$. Now consider a guessing game between two agents in which one agent thinks of a number z, and the other agent, a, must guess what this number is. We can assume the range of numbers that z is in is large but finite, say $3^{\mathfrak{n}}$ for a sufficiently large $\mathfrak{n}$, and the guesser thinks that each number is equally likely. Let each world in a model of CPL represent a possible value of z, therefore in a normalized model the guesser assigns a lottery of $\frac{1}{3^{\mathfrak{n}}}$ to each world. For each world $1 \leq q \leq 3^{\mathfrak{n}}$ there is a unique proposition p_q such that p_q says q is the correct number and for all other $j \in [1, 3^{\mathfrak{n}}]$, $\neg p_j$ holds. For a sufficiently large number $\mathfrak{n}$, the lottery of any given world is negligible since $\frac{1}{3^{\mathfrak{n}}} < \exp(-\mathfrak{n})$, and thus the probability a assigns to a proposition $\neg p_j$ for any $j \in [1, 3^{\mathfrak{n}}]$ is $1 - \frac{1}{3^{\mathfrak{n}}}$. Therefore, for every number $j \in [1, 3^{\mathfrak{n}}]$ we can say in CPL that $K_a \neg p_j$.

The lottery paradox states that it is rational to take the conjunction of these propositions $\neg p_j$ and state that the agent a would then know that no number in $[1, 3^{\mathfrak{n}}]$ is the correct number, which contradicts an initial assumption about the game, namely that there is a correct number to be guessed. It is easy to check in our example, that the conjunction $\neg p_1 \wedge \ldots \wedge \neg p_{3^{\mathfrak{n}}}$ will not hold in any world, and thus the probability that a will assign to such a conjunction is 0. Despite the guesser a *knowing* (under the CPL definition of knowledge) that for each number, that number is not the correct number, a never reaches the conclusion that there is *no* correct number. Thus, CPL avoids the lottery paradox. Clearly, this scenario does bring to a light a false instance of knowledge, however given that the alternative is paradox, such instances can be justified.

4.3.3 CPL Characterization of Zero-Knowledge

Recall the conditions of a zero-knowledge protocol are completeness, soundness, and zero-knowledge. Since a protocol is a sequence of actions, we can define an action model $\mathfrak{A}$ such that the actions contained within $\mathfrak{A}$ constitutes a zero-knowledge protocol iff certain conditions in the form of $\mathcal{L}_{CPL}$ formulas hold. We can design formulas for completeness, soundness, and zero-knowledge within the language of $\mathcal{L}_{CPL}$ that are dependent on the execution of a sequence of actions in $\mathfrak{A}$.

Consider a zero-knowledge proof of a statement. Let a zero-knowledge protocol take place in a model with at least two agents where we can refer to the agents as prover P and verifier V. For simplicity in our meta-language, we can think of computational languages in **NP**, and say that our zero-knowledge protocols will prove that for a given language, some x is a member of that language. x will be a message in our *logical* language[6] $\mathcal{L}_{CPL}$. The zero-knowledge protocols themselves will aim to prove $\mathsf{has}_P y$, meaning that the prover in the protocol *has* a message y where y is the witness to the fact that x is in a specific computational language. For the following generalizations, we can let $\varphi := \mathsf{has}_P y$ for an arbitrary witness y. Since we will be checking whether a sequence of actions is a zero-knowledge protocol, we will assume a well-designed action model $\mathfrak{A}$ and say that $\mathfrak{A}$ is a zero-knowledge protocol for $\mathcal{M}$ iff the following formulas are satisfied in $\mathcal{M}$. Supposing z is a sufficiently large natural number, we can characterize completeness via the sentence:

$$\varphi \to [\mathfrak{A}]^z K_V \varphi$$

Where $[\mathfrak{A}]^z$ is the number of necessary applications of $\mathfrak{A}$ to $\mathcal{M}$ to reach a high level of certainty. A number z can be considered sufficiently large, in the case where updating a model $\mathcal{M}$ with some action model $[\mathfrak{A}]^z$ achieves the intended result of the protocol. Sufficiently large numbers are arbitrarily selected during the action model design phase and are simply a way to necessitate that the actions of the action model are used as intended. Recall that knowledge is an abbreviation for a high level of certainty. Thus, $K_V\varphi := \mathbb{P}_V\varphi = \alpha$ for some $\alpha \in (1-\epsilon(|x|), 1]$. Informally, the sentence states that if the prover has a valid witness, then after the execution of the protocol (which is an interaction with the prover), the verifier will have high certainty of the fact that the prover has the witness.

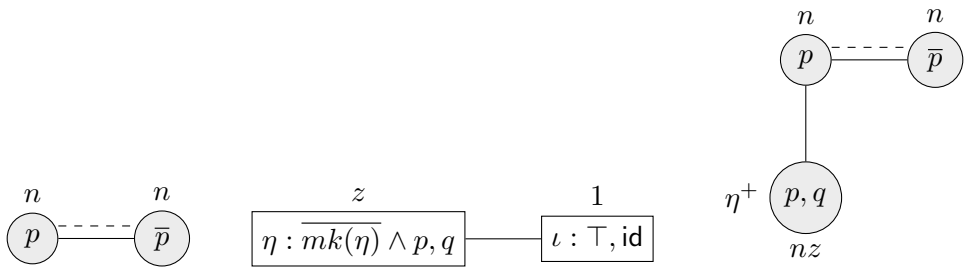
To get around the fact that the verifier shouldn't know the contents of the witness after the protocol, we can say that $\varphi := \mathsf{has}_P\{y\}_k$ for some encryption key k that is known only to the prover. Thus, by the cryptographic parsing function $\ulcorner\{y\}_k\urcorner_V^w = \blacksquare$ for all worlds w since the verifier will never know the key k. So the verifier can come to know that the prover has the witness, but cannot deduce its contents. As we will show, this will have no effect on the nature of the protocols or how the actions are executed, it is merely a semantic trick to prevent undesirable deductions in the logic.

The condition of soundness also must hold for a sufficiently large number z. The soundness condition will express that in the case where $\neg\varphi$, the probability that the verifier assigns to φ is less than a negligible number. That is, the probability where the verifier is successfully deceived by a dishonest prover is negligible. We can characterize soundness via the formal sentence:

$$\neg\varphi \to [\mathfrak{A}]^{z-1}\mathbb{P}_V(\langle\mathfrak{A},\sigma\rangle\top) < \beta$$

For some $\beta \in (0, \epsilon(|x|))$ where σ is an action executable in $\mathfrak{A}$ denoting a successful step in the protocol despite the prover not having a valid witness. The formula $\langle\mathfrak{A},\sigma\rangle\top$ allows the verifier

[6]Beware of the multiple uses of 'language'. I try to make it explicit when I am referring to a computational language in the sense of Turing machines and when I am referring to logical languages denoted by $\mathcal{L}$ (with some possible subscripts).



(a) Model $\mathcal{M}$ (b) Arbitrary Action Model $\mathfrak{S}_{\mathfrak{A}}$ (c) Update Model $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{A}}$ with copy of $\mathcal{M}$

Figure 4.3: Model Updates within $\mathfrak{S}_{\mathfrak{A}}$ such that the simulator has rewind capabilities.

to reason about the probability he assigns to a given world, by assigning a probability to the always true formula $\top$ in the world where a specific action can be executed. The reasoning for the structure of this formula will become clear when proving the soundness of a specific zero-knowledge protocol in Proposition 10. In both formulas for the soundness and completeness conditions, we can assume that z is the same number. Clearly, we can define $\mathfrak{A}$ as an interactive proof for a computational language L by the conjunction

$$(\varphi \to [\mathfrak{A}]^z K_V \varphi) \wedge (\neg\varphi \to [\mathfrak{A}]^{z-1} \mathbb{P}_V(\langle \mathfrak{A}, \sigma \rangle \top) < \beta)$$

where φ is a statement about the prover possessing a witness to a message $x \in L$.

Lastly, the condition of zero-knowledge states that the probability the verifier assigns to φ is indistinguishable from the probability a simulator would assign to φ had the simulator been given the same common input, auxillary information, and transcript of the communication. Such a simulated scenario would require another action model, called $\mathfrak{S}_{\mathfrak{A}}$ that is designed to simulate a dummy prover against the verifier's actions of $\mathfrak{A}$. In order to give the simulator rewind capabilities, we need to design the action model $\mathfrak{S}_{\mathfrak{A}}$ such that a copy of previous model updates is saved in current model updates, and at certain points the simulator can rewind the interaction by executing an action exclusively at the worlds that model the previous model update.

The simulator will require intentional design to ensure the lotteries of the previous model stay constant in the updated model. Furthermore, the probability the verifier assigns to a proposition should not be affected by access to the previous updates, so these copies of previous models will only be accessible to the simulator.

Let Sim be the simulated agent in the action model and V be the verifier. The actions of $\mathfrak{S}_{\mathfrak{A}}$ will follow a similar pattern to the actions of $\mathfrak{A}$ with the addition that certain actions are in the equivalence class $\sim_{Sim}(\iota)$. Those actions will be the actions which update the model with information we want to 'save' in the model. ι is an action with precondition $\top$ and postcondition id. The action ι has the lottery of 1 assigned to it, for all agents so the lotteries of the updated model, with respect to those worlds created by the ι action, are unchanged.

Figure 4.3 shows how such a simulator would function in an updated action model and how the rewind capabilities of the simulator are expressed in the model. We can let $\mathcal{M}$ be an arbitrary model with more than one world and lottery of n assigned to each world for both agents. The accessibility relation of the simulator Sim is denoted by the solid line and the accessibility relation of the verifier V is denoted by the dashed line. The action model $\mathfrak{S}_{\mathfrak{A}}$ contains an action, called η, with the precondition p and a postcondition that makes q true, as well as the ι action, both of which are in the same equivalence class of $\sim_{Sim}$. We also say that a precondition of η is that η is unmarked in the world, that is, it has not been executed there

before. The ι action has a lottery of 1 and the η action has a lottery of z. Note that the use of the precondition p in η is completely arbitrary and could have just as well been $\neg p$ or $\top$, but the goal of this example is to show that the initial model can be updated and restricted while also preserving a copy of the initial model.

The updated model in Figure 4.3(c) shows the initial model joined by Sim's accessibility relation to a world in which η has been executed. Notably, the lottery assigned to this new world is nz while the worlds that were in the initial model maintain their lotteries n. Furthermore, the accessibility relations of V are not altered by the addition of the ι action. The world where the η action was executed is labelled by the η^+ marking. It is possible for the simulator Sim to rewind the interaction, by executing the action model $\mathfrak{S}_{\mathfrak{A}}$ again on the model of Figure 4.3(c), in which the η action can only be executed in the world in the copy of the initial model where p holds, since η is unmarked there. Such an action (with the ι action as well) would result in an updated model that looks identical to Figure 4.3(c). Thus, the simulator was able to update the model $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{A}}$ with the model $\mathfrak{S}_{\mathfrak{A}}$ as if it was updating $\mathcal{M}$ with $\mathfrak{S}_{\mathfrak{A}}$.

Now that we are able to model a simulator with rewind capabilities, we can define what it means for an action model to be zero-knowledge. For a proposition $\varphi := \mathsf{has}_P y$, we want to express that the probability the verifier assigns to φ after an interactive protocol $\mathfrak{A}$ with P is indistinguishable from the probability that the verifier assigns to φ after an interactive protocol with a dummy prover in the simulation of $\mathfrak{S}_{\mathfrak{A}}$. This can be formalized in the language of $\mathcal{L}_{CPL}$ as follows:

$$\mathbb{P}_V([\mathfrak{A}]^z\varphi) \equiv \mathbb{P}_V([\mathfrak{S}_{\mathfrak{A}}]^{\rho(z)}\varphi)$$

where a statement of the form $\Lambda \equiv \Psi$ is an abbreviation of an expression for indistinguishability with respect to the absolute value of the difference between Λ and Ψ in the language of $\mathcal{L}_{CPL}$: $(\Lambda \leq \Psi \rightarrow (\Psi - \Lambda < \beta)) \vee (\Psi \leq \Lambda \rightarrow (\Lambda - \Psi < \beta))$ for some $\beta \in (0, \epsilon(n))$ for a sufficiently large n. The number of iterations $\rho(z)$ signifies that the iterations of $\mathfrak{S}_{\mathfrak{A}}$ are a function of the number of iterations of $\mathfrak{A}$.

Thus an action model $\mathfrak{A}$ is said to be a zero-knowledge proof for a computational language L if:

$$(\varphi \rightarrow [\mathfrak{A}]^z K_V \varphi) \wedge (\neg\varphi \rightarrow [\mathfrak{A}]^{z-1}\mathbb{P}_V(\langle\mathfrak{A}, \sigma\rangle\top) < \beta) \wedge \mathbb{P}_V([\mathfrak{A}]^z\varphi) \equiv \mathbb{P}_V([\mathfrak{S}_{\mathfrak{A}}]^{\rho(z)}\varphi)$$

where φ is a sentence about the prover's possession of some witness for $x \in L$.

The following section will analyze the exact construction of these action models and what each initial model will be composed of, as well as providing a demonstration of a zero-knowledge protocol for graph 3-colorability with a CPL model.

4.4 Zero-Knowledge Protocols

4.4.1 Modeling Protocols

A protocol will be a sequence of action patterns with specific pre and post-conditions of the form $a \rightarrow b : m$ where m is a message of a fixed pattern. The parameters of these patterns will have a fixed domain, and we say they are instantiated by an instantiation θ when parameters are mapped to objects in their respective domain. Unlike the models of Chapter 3 we will not assign roles as parameters of the protocols and instead leave the roles to be fixed as prover P and verifier V. This decision is not arbitrary, the protocols of Chapter 3 are flexible with respect to the roles of the agents so that they might alternate positions as sender and receiver in a cryptographic protocol to communicate with security, but for zero-knowledge protocols, the roles of prover and verifier are fixed in the sense that the protocol only needs to be executed in one direction.

Instantiations of a protocol can be interleaved in one run of the protocol, which as we will see in a coming example allows the agents of the protocol to make uniform decisions over the domains of the parameters. Therefore, the agents may repeat actions of the protocol, but with different instantiations mapping the parameters to the domain. The set of all possible instantiations is denoted as Θ, and since the set of parameters as well as the respective domains are finite, then Θ is finite [Dechesne and Wang, 2007].

As a further deviation from the models constructed in Chapter 3, we are not concerned with the possibility of an intruder eavesdropping on the communication. This arises from the nature of zero-knowledge protocols, where not enough information is transmitted for the verifier to learn the secret, let alone an intruder. In many ways, zero-knowledge protocols are intended to protect against the scenario in which the verifier is the intruder. Without this concern, we can simplify our models and subsequently our action models by bypassing any input and output buffers. Our actions can simply be transmitted directly from agent to agent.

As stated previously, our agents will be P and V to denote prover and verifier. An action of the form $P \to V : m$ states that P sends m to V, so we can design the precondition and postcondition accordingly. Without loss of generality, the default precondition states that $\mathsf{has}_P m$ holds in the world where the action is executed, and the postcondition $m \in I_V$ states that m is added to the information set of V in the updated model. Furthermore, the action becomes marked at the world in which it is executed. To tailor the protocol for specific action sequences, more elaborate preconditions and postcondtions can be used. As previously mentioned, for each action model $\mathfrak{A}$, there is an action model $\mathfrak{S}_{\mathfrak{A}}$ that simulates $\mathfrak{A}$ with a dummy prover agent Sim. Many of the actions remain the same bar the replacement of P with Sim, and there are additional actions ι, η_1, η_2 that are added. The ι action is as described previously, which allows the copies of the previous models to be saved within the updated models, and the η_1, η_2 actions are the actions which actually *rewind* the updated model to these saved models. As will be seen in the example protocol, the zero-knowledge aspect of the simulated action model arises in the internal computations made by Sim. Therefore, a transcript of $\mathfrak{A}$ and $\mathfrak{S}_{\mathfrak{A}}$, from V's perspective, are indistinguishable.

We now define an initial model for our protocols in a general sense to capture what necessary information must be implemented while simultaneously describing the initial model of a simulation, since each simulation will be derived from an actual initial model. Clearly the two agents are P and V for our initial model, while in parallel we know the agents of the simulation are Sim and V. The zero-knowledge protocol is concerned with a message x and whether x has some property. A message y is a witness to the fact that x has the property in question if y is an instance of x having this property. The point of a zero-knowledge proof is for the prover to prove that x has some property, so in an honest situation, it is the case that $\mathsf{has}_P y$ holds. Thus, our initial model is composed of the world where $\mathsf{has}_P y$ holds and the world where $\neg\mathsf{has}_P y$ holds. Naturally, the verifier cannot distinguish between these two worlds, otherwise the proof would be trivial. Furthermore, we assign to each agent the same lottery that maps both worlds to 1. This is a choice made for simplicity, so the probabilities the verifier assigns to each proposition will cleanly follow after an action. It is possible to have an initial model that is normalized, but it would require additional computational steps with respect to assigning probabilities after actions, so we use lotteries of 1 to avoid this. Since each world has a lottery of 1, then for any model update the lottery at the updated world simply becomes the lottery of the action that was executed there. The same holds for the initial models of the simulations.

The information sets of each agent in the initial model are simple to describe. For all agents $i \in A$ and worlds $w \in W$, $x \in I_{w,i}$ holds, which means that all agents have the common input of x. Furthermore, the distinction between the worlds is with respect to the prover's possession of the witness y, so in one world $y \in I_{w,P}$ and in the other world $y \notin I_{w',P}$. We can assume that

for all worlds w: $y \notin I_{w,V}$. Any further auxiliary information such as encryption keys or other messages can be implemented on a case by case basis at the needs of the protocol description. In a simulation initial model, both agents Sim and V possess x, but it's important to note that Sim does not possess y. In fact, the point of the simulation is to show that V enters a simulation with Sim and arrives at the same conclusion as he would have with P because of Sim's rewind capabilities, thus demonstrating zero-knowledge. Except for the witness y, Sim will be implemented with necessary auxiliary information such as encryption keys. In both the initial models and the simulated initial models, no actions are marked so $AL(w) = \emptyset$ for all worlds $w \in W$.

With the general description of initial models in place, we can analyze a zero-knowledge protocol description for proving whether a graph is 3-colorable. First, we will demonstrate how the necessary information of graph coloring can be encoded in the syntax of $\mathcal{L}_{CPL}$. Next, we will construct the action model and initial model of the protocol. Finally, we will verify within the logic that the action model satisfies completeness, soundness, and zero-knowledge.

4.4.2 Zero-Knowledge Protocol Example: Graph 3-Colorability

It is easy to see that one could represent a graph as a message in the language $\mathcal{L}_{CPL}$. By assigning a number to each vertex of the graph one could create a list of all vertices, and by joining these numbers in pairs one could represent an edge of the graph. We can encode a simple graph as seen in Figure 4.4, into the message $((1,(2,3)),((1,2),(1,3)))$. The first tuple contains the set of all vertices in the graph, and the second tuple contains the edges of the graph represented as pairs of vertices.

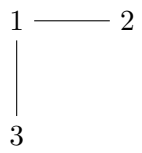


Figure 4.4: Simple Graph

Let $G(\mathbb{V}, E)$ be a graph with vertices $\mathbb{V}$ and edges E. If $G(\mathbb{V}, E)$ has a 3-color assignment, then it is possible to color each vertex of the graph with 3 distinct colors such that for every edge $(t, l) \in E$, the color of t is not identical to the color of l. Formally, $G(\mathbb{V}, E)$ is 3-colorable if there exists a mapping $\rho : \mathbb{V} \to \{r, b, g\}$ such that for every $(t, l) \in E : \rho(t) \neq \rho(l)$. For simplicity we let $\{r, b, g\}$[7] represent a set of distinct colors (red, blue, green). We can encode a coloring of a graph as a message in M, which will allow for relevant statements in the language of $\mathcal{L}_{CPL}$, by pairing a vertex number with a color, so a coloring for the graph in Figure 4.4 can be encoded as the concatenation of the pairs: $(1, r), (2, b), (3, g)$. Thus, if a prover has this message, then they have a witness to the fact that the graph $x = G((1,(2,3)),((1,2),(1,3)))$ is 3-colorable. In a message that is a witness, for a given vertex v, let $\mathsf{col}(v)$ denote the color assigned to that vertex.

Suppose a prover knows a 3-coloring of a graph $G(\mathbb{V}, E)$ and wishes to prove that fact to a verifier without revealing the coloring itself. At the end of the conversation the verifier will be convinced that there is a coloring of the graph satisfying the 3-color definition, yet he will not know that coloring. Let $c = |\mathbb{V}|$, $f = |E|$. Recall, a coloring is a map from vertices in the graph to a coloring in $Sym(\{r, b, g\})$, which are the permutations over the set $\{r, b, g\}$. So, $\rho \in Sym(\{r, b, g\})$. Let π also be a permutation over the set $\{r, b, g\}$, then for a vertex v_j,

[7]Recall, messages are numbers, but for readability let r, b, g be some numbers not in $\mathbb{V}$.

$\pi(\rho(v_j))$ is $\pi \circ \rho(v_j)$. With such a permutation and an encoding of the graph and coloring in the language $\mathcal{L}_{CPL}$, the prover and the verifier can use the following protocol description:

1. Repeat $c \times f$ times:
   a) $P \to V : (\{\pi(\mathsf{col}(v_1))\}_{k_1}, ..., \{\pi(\mathsf{col}(v_c))\}_{k_c})$
   b) $V \to P : (v_t, v_l) \in E$
   c) $P \to V : (k_t, k_l)$

The above protocol works by having the prover use a permutation on the set of colors and apply that to the coloring that she knows of the graph. With all vertex colors permuted, she then encrypts each color with an *individual key* associated with each vertex[8], thus preventing someone from learning one key and gaining access to all the vertices. The prover sends these encrypted permutations to the verifier. The verifier then chooses an edge in the graph and sends this edge back to the prover. The prover then sends the keys that decrypt the colorings of the vertices in the edge. Lastly, the verifier checks whether the colors are identical in the edge selected. If not, then the protocol continues, else the verifier knows with certainty that the prover does not have a coloring to the graph.

4.4.2.1 Action Model

The concept of selection appears twice in the above protocol: first when the prover selects a permutation and second when the verifier selects an edge to send back to the prover. When designing the action model, we can set these two items to be parameters such that they are instantiated at every iteration of the action model and thus we can get a uniform distribution over the domain of the parameters by randomly selecting an instantiation $\theta \in \Theta$. With this in mind, we can design the action model $\mathfrak{Z}$ and specify the action patterns to model the protocol description.

Figure 4.1 lists the actions of $\mathfrak{Z}$ for an overview of the protocol. To fit the table within the margins, let $\psi := (\{\pi(\mathsf{col}(v_1))\}_{k_1}, ..., \{\pi(\mathsf{col}(v_c))\}_{k_c})$, and furthermore, let us assume that each action has the precondition that the sender *has* the message being sent in conjunction with the other formulas listed in the Pre column of the table. The addition of the $\mathbb{L}$ column denotes the lottery each action has in $\mathfrak{Z}$. Since we are only concerned with the probabilities that the verifier assigns to actions, and because P has no accessibility relations between actions other than reflexive ones, we can say that the lottery listed in the $\mathbb{L}$ column is applied to both agents in the model.

Notably, the σ and δ actions of $\mathfrak{Z}$ both have a lottery of 1. This is because the contents of these messages do not lead to any ambiguity and their execution has no bearing on the assignment of probabilities. However, both of these actions utilize the θ instantiation within their message. For the σ action, the θ maps the parameter of π to a permutation in $Sym(\{r, b, g\})$, and for the δ action, the θ maps the parameter of t and l to numbers such that $1 \leq t, l \leq c$. The θ instantiation is only applied to these actions, so for actions $\gamma_{\{1,2,3\}}$ the t and l indexes are the same t, l that were set in the δ action. In that sense, the $\gamma_{\{1,2,3\}}$ actions are dependent on the δ action instantiation.

The set of actions $\gamma_{\{1,2,3\}}$ represent the three possible situations that can arise from the prover's response. γ_1 is the case where the prover has the witness y and sends the corresponding keys to the edge selected by the verifier. The lottery that a prover sends a set of keys that decrypts two non-identical vertex colors and has the witness y is 1. Meanwhile, in the case

[8]This encryption is meant to parallel the commitment function of the computational model from Section 4.2.1. For other attempts at modeling commitment protocols in epistemic logics see: Yu [2004], Hadzilacos [1987]

Action	Direction	Message	Pre	Pos_I	Pos_{AL}	$\mathbb{L}$
σ	$P \to V$	$(\theta)\psi$	$\neg mk(\sigma)$	$\psi \in I_V$	σ^+	1
δ	$V \to P$	$(\theta)(v_t, v_l)$	$mk(\sigma) \wedge \neg mk(\delta)$	$v_t, v_l \in I_P$	δ^+	1
γ_1	$P \to V$	(k_t, k_l)	$mk(\delta) \wedge \mathsf{has}_P y$	$k_t, k_l \in I_V$	σ^-, δ^-	1
γ_2	$P \to V$	(k_t, k_l)	$mk(\delta) \wedge \neg\mathsf{has}_P y$	$k_t, k_l \in I_V$	-	$\frac{1}{f}$
γ_3	$P \to V$	(k_t, k_l)	$mk(\delta) \wedge \neg\mathsf{has}_P y$	$k_t, k_l \in I_V$	σ^-, δ^-	$\frac{f-1}{f}$

Table 4.1: Action Model $\mathfrak{Z}$ for a zero-knowledge protocol of graph 3-colorability

where the prover does not have a witness y and sends the keys to the selected edge, then there is a minimum lottery of $\frac{1}{f}$ that the verifier will deduce that $\neg\mathsf{has}_P y$ holds. Therefore, in the same case the prover without the witness sends the keys of the edge, there is a maximum lottery of $\frac{f-1}{f}$ that the verifier will decrypt two non-identical vertex colors. Structuring the lotteries of the actions in this way will allow the model updates to depict the three possible scenarios in the protocol with an accurate probability assigned to each possibility. Furthermore, all actions of $\mathfrak{Z}$ are distinguishable from one another except for γ_1 and γ_3, which are both in the same equivalence class of $\sim_V$. This represents the verifier's inability to distinguish whether or not the prover has a valid coloring or has just gotten lucky during the decryption step of the protocol.

The preconditions and action postconditions are written in such a way, that it guarantees iterated model updates will follow a specific pattern. The σ action can only be executed if it is unmarked, and once it is marked, then the δ action can be executed. The δ action cannot be executed unless it is unmarked, and once it is marked, the $\gamma_{\{1,2,3\}}$ actions can be executed, which in turn unmark the σ and δ actions, allowing the cycle to repeat. The action γ_2 is an exception in that it does not unmark the σ and δ actions. This represents the immediate halting of the protocol in such a scenario where a pair of identically colored vertices is discovered.

4.4.2.2 Initial Model

With the action model $\mathfrak{Z}$ in place, we can establish the initial model and see how the model updates would affect the epistemic states of the agents. The initial model $\mathcal{M} = (W, R, I, AL, \mathbb{L})$ will consist of two worlds, both labeled by the proposition which holds in the world: $\mathsf{has}_P y$ and $\neg\mathsf{has}_P y$. Both worlds are joined by the equivalence relation of V. As previously stated, the lottery for each world is 1. Both P and V will know the common input of $x = G(\mathbb{V}, E)$, but P will know a witness y and a series of keys to encrypt each vertex in every round of the protocol. Thus, for $c \times f$ rounds, P will need $c \times (c \times f)$ symmetric encryption keys, one for each vertex in each round. The parameters of the protocol are π, t, l where the domain of π is $Sym(\{r, b, g\})$ and the domain of t, l is $\{n \mid v_n \in \mathbb{V}\}$. So the information states of both agents in all worlds $w \in W$ are set in the initial model as:

$$\{x\} \subseteq I_{w,V} \subset \{k_1, ...k_{c\cdot(c\cdot f)}, y\} \cup \{x\} \subseteq I_{w,P}$$

All actions are unmarked at all worlds in the initial model. Figure 4.5 illustrates a probability model for the initial model $\mathcal{M}$. Updating the model $\mathcal{M}$ with $\mathfrak{Z}$ does not result in any change to the model with respect to worlds and lotteries; it only changes in that at both worlds $\mathsf{has}_V \psi$ holds, where ψ is the constructed list of the encrypted, permuted colorings of vertices.

For z divisible by 3, $\mathcal{M} \circ \mathfrak{Z}^z$ results in a model of three worlds, each representing the possibilties expressed by the actions $\gamma_{\{1,2,3\}}$. Notably, the world in which the verifier discovers a pair of vertices with identical colors is distinguishable from the others. Figure 4.6 illustrates

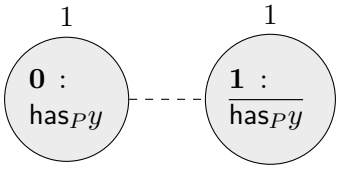


Figure 4.5: Initial Model $\mathcal{M}$ for zero-knowledge protocol

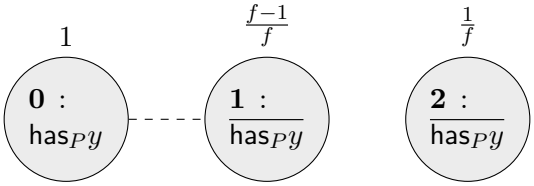


Figure 4.6: Update model $\mathcal{M} \circ \mathfrak{Z}^3$

the first of these update models: $\mathcal{M} \circ \mathfrak{Z}^3$ assigns a lottery of 1 at world **0** in which $\mathsf{has}_P y$ holds. This world is indistinguishable from world **1** where $\neg\mathsf{has}_P y$ holds, the lottery assigned to **1** is $\frac{f-1}{f}$. The disconnected world **2** also satisfies $\neg\mathsf{has}_P y$, but has a lottery of $\frac{1}{f}$. In worlds **0** and **1**, $\pi(\mathsf{col}(v_t)) \neq_V \pi(\mathsf{col}(v_l))$ holds because $\ulcorner\{\pi(\mathsf{col}(v_t))\}_{k_t}\urcorner_V^{\mathbf{0}} \neq \ulcorner\{\pi(\mathsf{col}(v_l))\}_{k_l}\urcorner_V^{\mathbf{0}}$ and likewise for **1**. World **2** represents the situation that satisfies $\pi(\mathsf{col}(v_t)) =_V \pi(\mathsf{col}(v_l))$.

At this stage in the protocol, in worlds **0** and **1**, the verifier assigns probability $\mathbb{P}_V(\mathsf{has}_P y) = \frac{1}{1+\frac{f-1}{f}}$. Clearly, as $\lim_{f\to\infty} \frac{1}{1+\frac{f-1}{f}} = \frac{1}{2}$. Likewsie, $\mathbb{P}_V(\neg\mathsf{has}_P y) = \frac{f-1}{2f-1}$ and as $\lim_{f\to\infty} \frac{f-1}{2f-1} = \frac{1}{2}$. Generally, this follows our intuition about the zero-knowledge protocol, in that at the beginning stages the verifier is uncertain about whether or not the prover has the witness. To continue updating the model with $\mathfrak{Z}^{3z}$ would result in updating the lotteries assigned to the worlds **0** and **1** such that $\mathbb{P}_V(\mathsf{has}_P y) = \frac{1}{1+\left(\frac{f-1}{f}\right)^z}$ and $\mathbb{P}_V(\neg\mathsf{has}_P y) = \frac{(f-1)^z}{f^z+(f-1)^z}$. The probability assigned to the world **1** can be thought of as the probability of a prover successfully cheating the verifier. Thus for z rounds of the protocol and subsequently $3z$ model updates, as $\lim_{z\to\infty} \mathbb{P}_V(\neg\mathsf{has}_P y) = 0$ which means as the number of rounds of the protocol in which the verifier decrypts the vertices and sees non-identical colors increases, the verifier's certainty level that the prover has a valid witness increases. As the protocol stipulates, the action model should update $\mathcal{M}$ at least $3 \times (c \times f)$ times which we can denote as $3u$.

4.4.2.3 Verification

Proposition 9 (Completeness of $\mathfrak{Z}$)**.** *Let y be a witness for the fact that x has the property of being 3-colorable, then following is true in the initial model $\mathcal{M}$:*

$$\mathsf{has}_P y \to [\mathfrak{Z}]^{3u} K_V \mathsf{has}_P y$$

Proof. Recall $K_i\varphi := \mathbb{P}_i\varphi = \alpha$ for some $\alpha \in (1-\epsilon(n), 1]$ for a sufficiently large n, and in the case of a zero-knowledge protocol for x, $\alpha \in (1-\epsilon(|x|), 1]$. Let w be a world in $\mathcal{M}$ such that

$\mathcal{M}, w \vDash \mathsf{has}_P y$. The only world in $\mathcal{M}$ where this holds is world $\mathbf{0}$. In $\mathcal{M} \circ \mathfrak{Z}^{3u}$ it holds that

$$\begin{aligned}
\mathbb{P}_{V,\mathbf{0}}^{\mathcal{M}\circ\mathfrak{Z}^{3u}}(\mathsf{has}_P y) &= \frac{\mathbb{L}_V(\mathbf{0})}{\mathbb{L}_V(\mathbf{0}) + \mathbb{L}_V(\mathbf{1})} \\
&= \frac{1}{1 + \left(\frac{f-1}{f}\right)^u} \\
&= \alpha
\end{aligned}$$

Now, we want to show that $\alpha \in (1 - \epsilon(|x|), 1]$. We assume x has at least 3 vertices, and since $u \in \mathbb{N}$ by Definition 39, then clearly $\alpha \leq 1$.

Since x is a simple and connected graph, we can show that c and f are polynomially related in that $c - 1 \leq f < \frac{c^2}{2}$ [Goldreich et al., 1991] and by assumption $u = c \times f$. Therefore there is some polynomial function h such that $h(|x|) = u$. By taking the limit of $\left(\frac{f-1}{f}\right)^u$ as f approaches ∞ we will show that $\left(\frac{f-1}{f}\right)^u$ is approximately equivalent to the negligible expression $\exp(-c)$.

$$\begin{aligned}
\Delta &= \lim_{f\to\infty} \left(\frac{f-1}{f}\right)^u \\
&= \lim_{f\to\infty} \left(\frac{f-1}{f}\right)^{c\cdot f}
\end{aligned}$$

By taking the natural log of both sides, we obtain:

$$\begin{aligned}
\ln \Delta &= \lim_{f\to\infty} c \cdot f \ln\left(\frac{f-1}{f}\right) \\
&= \lim_{f\to\infty} \frac{\ln\left(\frac{f-1}{f}\right)}{\frac{1}{cf}}
\end{aligned}$$

We can then use L'Hôpital's Rule, which states $\lim_{a\to b} \frac{j(a)}{g(a)} = \lim_{a\to b} \frac{\frac{d}{da} j(a)}{\frac{d}{da} g(a)}$, to solve for the limit [L'Hospital, 1696].

$$\begin{aligned}
&= \lim_{f\to\infty} \frac{\frac{d}{df} \ln\left(\frac{f-1}{f}\right)}{\frac{d}{df} \frac{1}{cf}} \\
&= \lim_{f\to\infty} \frac{\frac{1}{f(f-1)}}{-\frac{1}{cf^2}} \\
&= \lim_{f\to\infty} -\frac{cf}{f-1}
\end{aligned}$$

As f approaches ∞, this rational becomes $-c$, therefore we have:

$$\begin{aligned}
\ln \Delta &= -c \\
\Delta &= \exp(-c) = e^{-c} = \frac{1}{e^c}
\end{aligned}$$

Thus, for a sufficiently large graph x such that $h(|x|) = u$, Δ is negligible and $\alpha \in (1-\Delta, 1]$, so this implies that $\alpha \in (1-\epsilon(|x|), 1]$ for a negligible function ϵ on the size of x. □

Recall, the concept of soundness in zero-knowledge protocols refers to the probability a verifier accepts a statement φ as valid, even when $\neg\varphi$ is true. This can be represented in the logical protocol as the probability that the verifier is successfully tricked by a dishonest prover. That is, the lottery of the world where γ_3 is executed is the probability that the verifier is successfully deceived. We can reason about this lottery by taking the probability that the verifier assigns to the formula $[\mathfrak{Z}]^{3u-1}\langle\mathfrak{Z}, \gamma_3\rangle\top$.

Proposition 10 (Soundness of $\mathfrak{Z}$)**.** *Let y be a witness for the fact that the graph x has the property of being 3-colorable, then the following holds in $\mathcal{M}$:*

$$\neg\mathsf{has}_P y \to [\mathfrak{Z}]^{3u-1}\mathbb{P}_V(\langle\mathfrak{Z}, \gamma_3\rangle\top) < \beta$$

for some $\beta \in (0, \epsilon(|x|))$.

Proof. Let w be a world in $\mathcal{M}$ such that $\mathcal{M}, w \vDash \neg\mathsf{has}_P y$. The only world in $\mathcal{M}$ where this holds is world $\mathbf{1}$. Update the model $\mathcal{M}$ with $\mathfrak{Z}^{3u-1}$ and we want to show that the probability V assigns to being in a world where the γ_3 actions has been executed is negligible.

$$\begin{aligned}\mathbb{P}_{V,\mathbf{1}}^{\mathcal{M}\circ\mathfrak{Z}^{3u-1}}(\langle\mathfrak{Z}, \gamma_3\rangle\top) &= \frac{\sum\{\mathbb{L}_V(z) \mid \langle w, z\rangle \in R_V \,\&\, \mathcal{M}\circ\mathfrak{Z}^{3u}, \langle z, \gamma_3\rangle \vDash \top\}}{\sum\{\mathbb{L}_V(z) \mid \langle w, z\rangle \in R_V\}} \\ &= \frac{\left(\frac{f-1}{f}\right)^u}{1+\left(\frac{f-1}{f}\right)^u}\end{aligned}$$

By Proposition 9, we know this is equivalent to:

$$\mathbb{P}_{V,\mathbf{1}}^{\mathcal{M}\circ\mathfrak{Z}^{3u-1}}(\langle\mathfrak{Z}, \gamma_3\rangle\top) = \frac{\Delta}{1+\Delta}$$

where Δ is a negligible function on the size of x. We can show that $\frac{\Delta}{1+\Delta} \leq \Delta$:

$$\begin{aligned}\frac{\Delta}{1+\Delta} &\leq \Delta \\ \Delta &\leq \Delta(1+\Delta) \\ \Delta &\leq \Delta + \Delta^2\end{aligned}$$

The truth of which immediately follows from the fact that $\Delta > 0$. Therefore, we can show that $\mathbb{P}_{V,\mathbf{1}}^{\mathcal{M}\circ\mathfrak{Z}^{3u-1}}(\langle\mathfrak{Z}, \gamma_3\rangle\top) < \beta$ for $\beta \in (0, \epsilon(|x|))$ and thus, $\mathfrak{Z}$ is a sound protocol. □

Corollary 1. *It follows immediately from Propostion 9 and Proposition 10 that the action model $\mathfrak{Z}$ is an interactive protocol for the language of 3-colorable graphs.*

The simulation for the protocol will work in a way that the simulator does not need to know a valid 3-coloring of the graph x. Rather, at the beginning of each iteration of the protocol, the simulator will select an edge (v_e, v_d) from the graph and assign a coloring to this edge such that $\mathsf{col}(v_e) \neq \mathsf{col}(v_d)$ The simulator will then assign a null color 0 to all other vertices in the graph. After encrypting each coloring for the vertices and sending the list of encrypted

messages to the verifier, the verifier selects an edge in the graph and sends it to the simulator. If the edge selected by the verifier is the edge that the simulator assigned a non-null coloring to, then the protocol proceeds as normal where the simulator sends the keys corresponding to the encrypted messages which contain those colorings. If the edge selected is not the edge the simulator assigned a non-null coloring to, then the simulator rewinds the interaction to the step in which the simulator selects an edge (v_e, v_d). From this point the iteration continues as before.

To prove that $\mathfrak{Z}$ is also a zero-knowledge protocol, we must design the action model $\mathfrak{S}_{\mathfrak{Z}}$ which will replace the prover in $\mathfrak{Z}$ with Sim in $\mathfrak{S}_{\mathfrak{Z}}$. The set of actions in $\mathfrak{S}_{\mathfrak{Z}}$ contains the ι action, so as the model is being updated with respect to the θ instantiation made in the σ and δ actions, copies of the previous updates are saved within the updated model. In order to keep the information saved by the ι action restricted to the information created during the selection of $\theta \in \Theta$, $\sigma \sim_{Sim} \iota$ and $\delta \sim_{Sim} \iota$ holds in $\mathfrak{S}_{\mathfrak{Z}}$. Thus, when the σ action is executed, a copy of the initial model is saved within the updated model $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$, and from an updated model $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$ when the δ action is executed a copy of the updated model $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$ is accessible to Sim in $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}{}^2$.

Once the verifier has made the selection of an edge (v_t, v_l) and sent it to the simulator, if $t = c$ and $l = d$, then $\ulcorner\{\mathsf{col}(v_t)\}_{k_t}\urcorner^w_{Sim} \neq \ulcorner\{\mathsf{col}(v_l)\}_{k_l}\urcorner^w_{Sim}$ for all $w \in W$, with the appropriate indices attached to k in both instances. Rather than rewind, the simulator proceeds, but in the action model we must now disconnect the *saved* information since the simulator will not need it. This is done by executing the η_2 action which requires that $\ulcorner\{\mathsf{col}(v_t)\}_{k_t}\urcorner^w_{Sim} \neq \ulcorner\{\mathsf{col}(v_l)\}_{k_l}\urcorner^w_{Sim}$ holds as well as $mk(\sigma)$ and $mk(\delta)$, thus ensuring that the new model represents the current state of the iteration where the keys can be sent from the simulator to the verifier. The action η_2 has the postcondition of id which does not change any agent's information state. The simulator can then send the keys of the colorings of those vertices. The verifier then checks whether $\mathsf{col}(v_t), \mathsf{col}(v_l) \in \{r, b, g\}$ and whether $\mathsf{col}(v_t) \neq_V \mathsf{col}(v_l)$ holds as in the standard action model $\mathfrak{Z}$.

If the edge selected by the verifier is one of the edges that received a null coloring by the simulator such that $(v_t, v_l) \neq (v_e, v_d)$ then $\ulcorner\{\mathsf{col}(v_t)\}_{k_t}\urcorner^w_{Sim} \neq \ulcorner\{\mathsf{col}(v_l)\}_{k_l}\urcorner^w_{Sim}$ holds in all $w \in W$. To rewind the interaction, which is essentially resetting the model to a previous instance, the η_1 action is executed, which requires that $\ulcorner\{\mathsf{col}(v_t)\}_{k_t}\urcorner^w_{Sim} \neq \ulcorner\{\mathsf{col}(v_l)\}_{k_l}\urcorner^w_{Sim}$ holds as well as $\neg mk(\sigma)$ which ensures that the model is reset to an update where $(\theta)\psi$ has not been instantiated yet. From this point, the iteration will continue as normal with the simulator selecting an instantiation $\theta \in \Theta$ which selects an edge (v_e, v_d) for the simulator to assign a coloring to.

Action	**Direction**	**Message**	Pre	Pos_I	Pos_{AL}	$\mathbb{L}$
σ	$Sim \to V$	$(\theta)\psi$	$\neg mk(\sigma)$	$\psi \in I_V$	σ^+	1
δ	$V \to Sim$	$(\theta)(v_t, v_l)$	$mk(\sigma) \wedge \neg mk(\delta)$	$(v_t, v_l) \in I_{Sim}$	δ^+	1
η_1	-	-	$\neg mk(\sigma) \wedge \mathsf{col}(v_t) =_{Sim} \mathsf{col}(v_l)$	id	-	1
η_2	-	-	$mk(\sigma) \wedge mk(\delta) \wedge \mathsf{col}(v_t) \neq_{Sim} \mathsf{col}(v_l)$	id	-	1
γ_1	$Sim \to V$	(k_t, k_l)	$mk(\sigma) \wedge mk(\delta) \wedge \mathsf{has}_{Sim} y$	$(k_t, k_l) \in I_V$	σ^-, δ^-	1
γ_2	$Sim \to V$	(k_t, k_l)	$mk(\sigma) \wedge mk(\delta) \wedge \neg\mathsf{has}_{Sim} y$	$(k_t, k_l) \in I_V$	σ^-, δ^-	$\frac{f-1}{f}$
ι	-	-	$\top$	id	-	1

Table 4.2: Description of the action model $\mathfrak{S}_3$

Table 4.2 lists the actions of $\mathfrak{S}_{\mathfrak{Z}}$ with the corresponding descriptions and requirements. The actions η_1, η_2, ι notably do not have a direction nor a message, this is because these actions are intended to manipulate the model itself and not represent a proccess of communication in the protocol. The lack of message and direction is reflected in the postcondition of id. The specific precondition formulas ensure that the actions of $\mathfrak{S}_{\mathfrak{Z}}$ are executed in a specific cyclic order similar to $\mathfrak{Z}$ and that the actions η_1 and η_2 are only executed under certain circumstances, namely when the edge selected in the δ action is the same edge selected in the σ action.

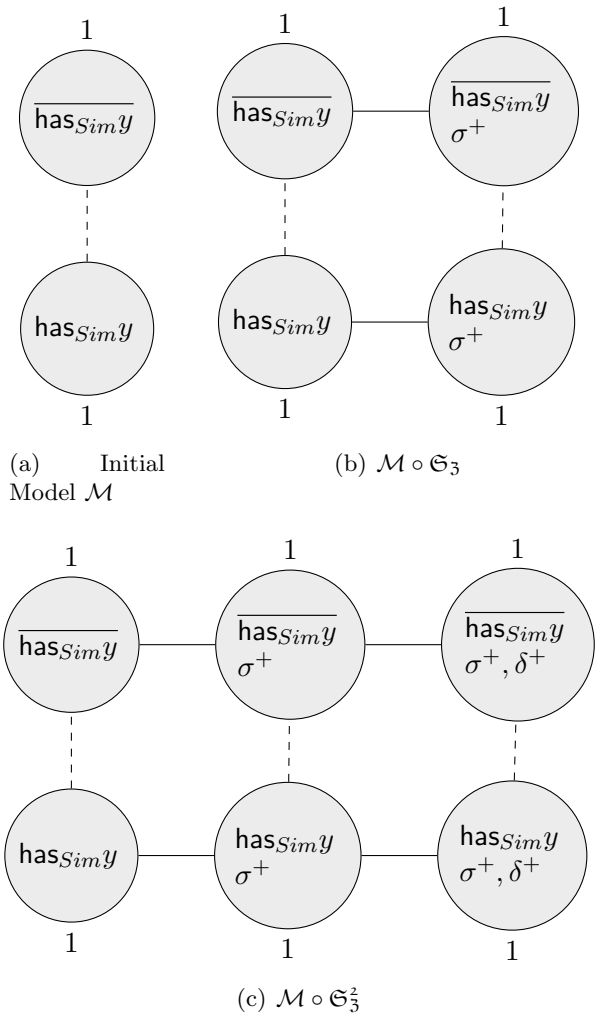
All actions in $\mathfrak{S}_{\mathfrak{Z}}$ have a lottery of 1 except for the action γ_2, since γ_2 represents the action in which a possible prover in the protocol would supply the keys to an encrypted pair of vertices that happen to have non-identical colorings even though the prover does not know a valid coloring for the graph x. Thus, this event occurs with a maximum probability of $\frac{f-1}{f}$. Since the simulator has rewind capabilities, we do not consider the event in which the simulator sends a pair of keys to decrypt a pair of identical vertex colorings.

Epistemically, the accessibility relations $\sim$ in $\mathfrak{S}_{\mathfrak{Z}}$ are constructed to make the model updates compatible with the rewind capabilities of the simulator. Therefore, V can distinguish between all actions in $\mathfrak{S}_{\mathfrak{Z}}$ except for γ_1 and γ_2. So $\sim_V = \{\langle\lambda,\lambda\rangle \mid \lambda \in \mathfrak{S}_{\mathfrak{Z}}\} \cup \{\langle\gamma_1,\gamma_2\rangle, \langle\gamma_2,\gamma_1\rangle\}$. The simulator Sim cannot distinguish between the ι action and the σ, δ actions which allows Sim to save the worlds updated by these actions within the model. Thus, $\sim_{Sim} = \{\langle\lambda,\lambda\rangle \mid \lambda \in \mathfrak{S}_{\mathfrak{Z}}\} \cup \{\langle\iota,\sigma\rangle, \langle\sigma,\iota\rangle, \langle\iota,\delta\rangle, \langle\delta,\iota\rangle, \langle\sigma,\delta\rangle, \langle\delta,\sigma\rangle\}$. For the other actions in $\mathfrak{S}_{\mathfrak{Z}}$, Sim can distinguish between them.

Figure 4.7 illustrates a sequence of model updates starting with the initial model (Figure 4.7(a)) and first executing the σ action (Figure 4.7(b)), and then executing the δ action (Figure 4.7(c)). We let the dashed lines represent the accessibility relations of V, and the solid lines to represent the accessibility relations of Sim. To further distinguish which worlds are copies, we have labeled the worlds according to the action labels at the given world, so a world labeled with σ^+ denotes that σ is marked at that world. The initial model is unlabeled since no actions are marked there. As can be seen in Figure 4.7(b), a copy of $\mathcal{M}$ is accessible to Sim and likewise in Figure 4.7(c), a copy of $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$ is accessible to Sim.

In the model depicted in Figure 4.7(c), the next possible action depends on the instantiations θ used in the actions of σ and δ. Figure 4.8 shows the possible scenarios that can follow from the model in Figure 4.7(c). If the edge selected by the verifier and sent in the δ action was not the same edge selected by the simulator, then the updated model $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^3$ will be a model in which the η_1 action was executed, depicted in Figure 4.8(a). Notice, this model has no action labels, therefore it is a way for the simulator to rewind back to the initial model. If both the simulator and the verifier select the same edge, then the updated model $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^3$ will be a model where η_2 has been executed, an illustration can be found in Figure 4.8(b). The action labels denote that from this model the protocol can proceed with the message containing the keys to the encrypted colorings. Finally, Figure 4.8(c) depicts a model update from the model $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^3$ where η_2 was executed to a model where γ_1 and γ_2 have been executed. Notably, the lotteries assigned to each world in the model have changed since with each successful key delivery, the probability that the verifier assigns to the possible world where the prover does not have a valid witness decreases.

With a functioning simulation described via model updates, we can prove that $\mathfrak{Z}$ is a zero-knowledge protocol, since the probability that the verifier assigns to a prover having a valid witness y after an update of $\mathfrak{Z}^u$ will be indisitinguishable from the probability that the verifier assigns to the simulator having a valid witness y after an update of ${\mathfrak{S}_{\mathfrak{Z}}}^{\rho(u)}$ where $\rho(u)$ is some polynomial function on $u = c \times f$. Since a single iteration of the protocol takes at least 4 model updates, then the maximum number of model updates we want to allow for is $12 \times u^2 = 3u \times 4u$, accounting for the number of times we allow the simulator to rewind per iteration: u, where



(a) Initial Model $\mathcal{M}$

(b) $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$

(c) $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}^{z}$

Figure 4.7: Simulation of a zero-knowledge protocol for graph 3-colorability with $\mathfrak{S}_{\mathfrak{Z}}$ with a possible rewind action.

each rewind requires 3 model updates.

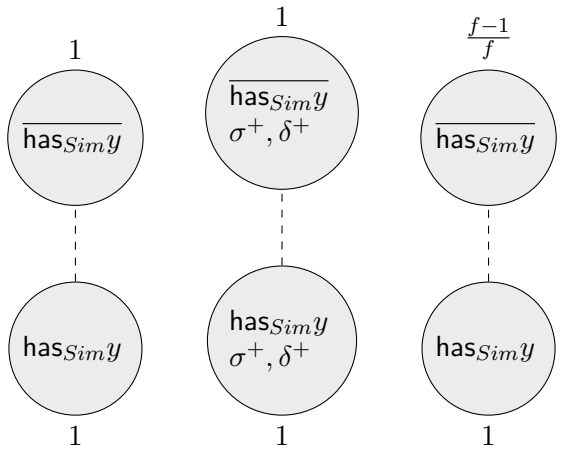
Proposition 11 ($\mathfrak{Z}$ Zero-Knowledge)**.** *Let y be a valid witness to the fact that graph x is 3-colorable. Let $u = h(|x|)$ for some polynomial function h on the size of x. The following is true in $\mathcal{M}$:*

$$\mathbb{P}_V([\mathfrak{Z}]^{3u}\mathsf{has}_P y) \equiv \mathbb{P}_V([\mathfrak{S}_{\mathfrak{Z}}]^{12u^2}\mathsf{has}_{Sim} y)$$

Proof. The probability the verifier V assigns to $[\mathfrak{Z}]^{3u}\mathsf{has}_P y$ is the probability that the verifier assigns to $\mathsf{has}_P y$ in the model $\mathcal{M} \circ \mathfrak{Z}^{3u}$, so we can solve for the probability with the following equation[9] from Definition 39:

$$\frac{\sum\{\mathbb{L}_V(z) \mid \langle w, z\rangle \in R_V \,\&\, \mathcal{M} \circ \mathfrak{Z}^{3u}, \langle z, \gamma_1\rangle \vDash \mathsf{has}_P y\}}{\sum\{\mathbb{L}_V(z) \mid \langle w, z\rangle \in R_V\}} = \frac{1}{1 + \left(\frac{f-1}{f}\right)^u}$$

[9]Here we let z abbreviate the world that has been updated $3u - 1$ times, thus $\langle z, \gamma_1\rangle$ is the world z in the update model $\mathcal{M} \circ \mathfrak{Z}^{3u}$ where $\mathsf{Pre}(\gamma_1)$ holds.



(a) $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^3$ with η_1

(b) $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^3$ with η_2

(c) $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^4$ if η_2 was executed

Figure 4.8: The possible model updates of $\mathcal{M} \circ \mathfrak{S}_{\mathfrak{Z}}$ depending on whether the preconditions of η_1 or η_2 are satisfied.

As shown in Proposition 9, $\Delta = \left(\frac{f-1}{f}\right)^u$ is negligible as f approaches ∞, so it follows that as $\lim_{f\to\infty} \mathbb{P}_V([\mathfrak{Z}]^{3u}\mathsf{has}_P y) = \frac{1}{1+\Delta}$.

The probability that the verifier V assigns to $[\mathfrak{S}_{\mathfrak{Z}}]^{12u^2}\mathsf{has}_{Sim}y$ is the probability that V assigns to $\mathsf{has}_{Sim}y$ in the model $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^{12u^2}$. Thus we can use the following equation[10] to define the probability:

$$\frac{\sum\{\mathbb{L}_V(\upsilon) \mid \langle w, \upsilon\rangle \in R_V \,\&\, \mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^{12u^2}, \langle \upsilon, \gamma_1\rangle \vDash \mathsf{has}_{Sim}y\}}{\sum\{\mathbb{L}_V(\upsilon) \mid \langle w, \upsilon\rangle \in R_V\}} = \frac{1}{1+\left(\frac{f-1}{f}\right)^{\kappa}}$$

where $u \leq \kappa \leq u^2$. So we want to show that the difference between the two values is less than the negligible number Δ:

$$\begin{aligned}
\lim_{f\to\infty} \frac{1}{1+\left(\frac{f-1}{f}\right)^{u^2}} - \lim_{f\to\infty} \frac{1}{1+\left(\frac{f-1}{f}\right)^{u}} &< \Delta \\
\frac{1}{1+0} - \frac{1}{1+\Delta} &< \Delta \\
1 - \frac{1}{1+\Delta} &= \frac{\Delta}{1+\Delta}
\end{aligned}$$

By Proposition 10, we know that $\frac{\Delta}{1+\Delta} < \Delta$. Thus, $\mathbb{P}_V([{\mathfrak{S}_{\mathfrak{Z}}}^{12u^2}]\mathsf{has}_{Sim}y) - \mathbb{P}_V([\mathfrak{Z}]^{3u}\mathsf{has}_P y) < \beta$ where $\beta \in (0, \epsilon(|x|))$. So, the probability that V assigns to $\mathsf{has}_P y$ after the protocol of $\mathfrak{Z}$ is indistinguishable from the probability that V assigns to $\mathsf{has}_{Sim}y$ after the protocol of $\mathfrak{S}_{\mathfrak{Z}}$. □

[10] We similarly let υ abbreviate the world that has been updated $12u^2 - 1$ times, so $\langle \upsilon, \gamma_1\rangle$ is the world in $\mathcal{M} \circ {\mathfrak{S}_{\mathfrak{Z}}}^{12u^2}$ where $\mathsf{Pre}(\gamma_1)$ holds.

Corollary 2. *From Corollary 1 and Proposition 11 it follows that* $\mathfrak{Z}$ *is a zero-knowledge protocol for the language of 3-colorable graphs.*

We have shown that in an interaction where the prover has a witness y for the fact that the graph x satisfies 3-colorability, then the verifier assigns a probability of 1 to the proposition $\mathsf{has}_P y$. In the alternative case where a prover does not have a witness y for the fact that x satisfies 3-colorability, we have shown that the probability the verifier assigns to being successfully deceived is negligible after a sufficiently large number of rounds in the protocol. Furthermore, we have shown that the probability the verifier assigns to the prover having a witness y is indistinguishable from the probability that the verifier assigns to a simulated prover having a witness y, thus demonstrating that the protocol satisfies the condition of *zero-knowledge*.

CHAPTER 5

Conclusion

We examined multiple variants of epistemic logics to model different types of cryptographic protocols. We verified certain security properties about each of the protocols analyzed as well as investigating the strength of the adversary in one of the protocols. The initial cryptographic variant of DEL is capable of modeling symmetric key encryption and adversarial agents. We extended the construction rules to allow for asymmetric encryption keys, making it possible to model signature schemes. Using verification techniques, we designed formal sentences within the logic that allow us to check the validity of statements which ensured security properties of the protocol held.

By combining methods of cryptographic modeling and notions of probability logics, we developed CPL which can characterize the three requirements of zero-knowledge protocols. Finally, we implemented the zero-knowledge protocol for graph 3-colorings in CPL. By modeling the zero-knowledge protocol of the graph 3-coloring problem, we showed that our representation of the protocol within CPL is accurate with respect to probability assignments and superficial knowledge of individual agents. Furthermore, we showed that upon repeated iterations of the protocol, notions of completeness, soundness, and zero-knowledge held. The verification of completeness and soundness for the zero-knowledge protocol in CPL were a product of the semantics of the lottery assignments in updated models of CPL and intentional design of the formal sentences in the language of CPL that the probability is being assigned to. Lastly, the notion of zero-knowledge required the introduction of simulators in the logic CPL. The simulations provide the verifier with the ability to run a protocol against a simulated prover with rewind capabilities, thus ensuring that the entire protocol is executable without the verifier learning any information.

Our decision to use the graph 3-coloring problem as our application of CPL stems from the fact that graph 3-coloring is **NP**-Complete. Therefore, all zero-knowledge protocols for problems in **NP** reducible to the graph 3-coloring problem can be modeled and verified in CPL.

Further areas of research regarding the topics covered in this thesis are plentiful. We only sketch a few here to give useful guidelines for further research. The protocol examples analyzed in Chapter 3, focused mainly on secrecy and verification in the case of the signature scheme. Further security properties such as authentication, non-repudiation, anonymity, and deniable authentication can be analyzed by implementing relevant protocols in the logic.

A further study of CPL would involve a complete axiomatization. Dechesne and Wang conjectured that the cryptographic variant of DEL which they introduced could be axiomatized in $\mathcal{S}5$ models, however that problem remains open. Furthermore, EPL is axiomatized in [van Eijck and Schwarzentruber, 2014]. These provide an essential foundation for the axiomatization

of CPL, however it is important to keep in mind that the definition of knowledge in EPL equates knowledge with absolute certainty, while CPL uses *high levels* of certainty.

Variations of zero-knowledge protocols include concurrent protocols which can take place between multiple provers and a single verifier or *vice versa*. Research around concurrent zero-knowledge protocols [Dwork and Sahai, 1998, Dwork et al., 2004, Lin et al., 2010] has identified security weaknesses in protocol structures, such that if colluding dishonest verifiers concurrently query a prover, then they can strategically query the prover such that they gain information, thus compromising the secret information of the prover. CPL would be ideal to model such concurrent scenarios and to analyze potential information leakage because of its multi-agent environment and modular action models. The solutions to these attacks proposed by Dwork and Sahai involve timing constraints such that if the verifier takes more than β amount of time to respond to a prover's challenge then the prover terminates the protocol. Modeling timing constraints and other temporal aspects of a cryptographic protocol in CPL remains an open area of interest.

The field of computational model checking is constantly advancing, and given the reliance on *sufficiently large* numbers in cryptography, utilizing computational methods for model checking would provide an efficient method for verifying the requirements of cryptographic protocols in CPL. Model checkers for epistemic logics have already been devised [van Ditmarsch et al., 2012, van Eijck and Orzan, 2007, Gattinger and van Eijck, 2015], however they often require fine-tuning for specific systems.

Lastly, to broaden the scope of research, many areas of cryptography are transitioning to the so-called 'post-quantum' world. Post-quantum cryptography focuses on cryptographic protocols that aim to be secure against quantum computers. It is known that quantum algorithms exist which can solve difficult problems that are fundamental to cryptographic security [Schmidt, 2006]. An obvious intersection of study would be to investigate recent advancements in quantum epistemic logics [Baltag and Smets, 2011, Baltag et al., 2013] and to investigate whether they are suitable for modeling post-quantum cryptographic protocols.

Bibliography

S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2006.

M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. Cryptology ePrint Archive, Report 2007/289, 2007. `https://eprint.iacr.org/2007/289`.

M. Backes, M. Maffei, K. Pecina, F. Bendun, and E. Mohammadi. Symbolic malleable zero-knowledge proofs. In *Proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF '15)*, 2015.

J. C. Baeten. Models of Computation: Automata, Formal Languages and Communicating Processes. Unpublished, 2020.

A. Baltag and L. S. Moss. Logics for epistemic programs. *Synth.*, 139(2):165–224, 2004.

A. Baltag and S. Smets. Probabilistic dynamic belief revision. *Synth.*, 165(2):179–202, 2008.

A. Baltag and S. Smets. Quantum logic as a dynamic logic. *Synthese*, 179:285–306, 2011.

A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '98, page 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

A. Baltag, J. M. Bergfeld, K. Kishida, J. Sack, S. J. L. Smets, and S. Zhong. Quantum probabilistic dyadic second-order logic. In L. Libkin, U. Kohlenbach, and R. de Queiroz, editors, *Logic, Language, Information, and Computation*, pages 64–80, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

A. Baskar, R. Ramanujam, and S. P. Suresh. A Dolev-Yao model for zero knowledge. In A. Datta, editor, *Advances in Computer Science - ASIAN 2009. Information Security and Privacy, 13th Asian Computing Science Conference, Seoul, Korea, December 14-16, 2009. Proceedings*, volume 5913 of *Lecture Notes in Computer Science*, pages 137–146. Springer, 2009.

P. Blackburn, M. de Rijke, and Y. Venema. *Modal logic*. Cambridge tracts in theoretical computer science ; 53. Cambridge University Press, 2001.

L. Carroll. *Curiosa Mathematica: Pillow Problems*. 1895.

X. Chen and H. Deng. Efficient verification of cryptographic protocols with dynamic epistemic logic. *Applied Sciences*, 10:6577, 2020.

W. Commons. File:zkip alibaba2.png — wikimedia commons, the free media repository, 2020. URL https://commons.wikimedia.org/w/index.php?title=File:Zkip_alibaba2.png&oldid=455408739.

F. Dechesne and Y. Wang. Dynamic epistemic verification of security protocols: framework and case study. 2007.

F. Dechesne and Y. Wang. To know or not to know: epistemic approaches to security protocol verification. *Synth.*, 177(Supplement-1):51–76, 2010.

D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 442–457, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.

R. Fagin and J. Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41(2): 340–367, 1994. ISSN 0004-5411.

R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87(1–2):78–128, 1990. ISSN 0890-5401.

M. Fischer and L. Zuck. Relative knowledge and belief. Technical Report YALEU/DCS/TR-589, Yale University, 1987.

T. French, A. Gozzard, and M. Reynolds. A modal aleatoric calculus for probabilistic reasoning. In M. A. Khan and A. Manuel, editors, *Logic and Its Applications - 8th Indian Conference, ICLA 2019, Delhi, India, March 1-5, 2019, Proceedings*, volume 11600 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 2019a.

T. French, A. Gozzard, and M. Reynolds. Aleatoric dynamic epistemic logic for learning agents. In A. C. Nayak and A. Sharma, editors, *PRICAI 2019: Trends in Artificial Intelligence - 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part I*, volume 11670 of *Lecture Notes in Computer Science*, pages 433–445. Springer, 2019b.

M. Gattinger and J. van Eijck. Towards Model Checking Cryptographic Protocols with Dynamic Epistemic Logic. In *Proceedings LAMAS (LAMAS 2015)*, Turkey, 2015.

O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 171–185, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have Zero-Knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.

S. Goldwasser and S. Micali. Probabilistic encryption how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 365–377, New York, NY, USA, 1982. Association for Computing Machinery.

S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof system. *SIAM J. Comput.*, 18:186–208, 1989.

V. Hadzilacos. A knowledge theoretic analysis of atomic commitment protocols. In M. Y. Vardi, editor, *Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 23-25, 1987, San Diego, California, USA*, pages 129–134. ACM, 1987.

J. Halpern, Y. Moses, and M. Tuttle. A knowledge-based analysis of zero knowledge. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 132–147, New York, NY, USA, 1988. Association for Computing Machinery.

J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, PODC '89, page 103–118, New York, NY, USA, 1989. Association for Computing Machinery.

J. Y. Halpern, R. Pass, and V. Raman. An epistemic characterization of zero knowledge. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '09, page 156–165, New York, NY, USA, 2009. Association for Computing Machinery.

A. Hasib and A. Haque. A comparative study of the performance and security issues of AES and RSA cryptography. In *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 02*, ICCIT '08, page 505–510, USA, 2008. IEEE Computer Society.

J. Herzog. A computational interpretation of Dolev-Yao adversaries. *Theor. Comput. Sci.*, 340 (1):57–81, 2005.

J. Katz and Y. Lindell. *Introduction to Modern Cryptography (Chapman Hall/Crc Cryptography and Network Security Series)*. Chapman Hall/CRC, 2007.

B. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12, 2003. doi: 10.1023/A:1025050800836.

B. Kooi, J. Gerbrandy, and J. van Benthem. Dynamic update with probabilities. *Studia Logica*, 93, 2009.

S. Kramer. *Logical Concepts in Cryptography*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

H. E. Kyburg Jr. *Probability and the Logic of Rational Belief*. Wesleyan University Press, 1961.

G.-F.-A. d. .-. A. d. t. L'Hospital. *Analyse des infiniment petits, pour l'intelligence des lignes courbes*. A Paris, de l'Imprimerie royale. M.DC.XCVI, Bibliothèque nationale de France, 1696.

H. Lin, R. Pass, W.-L. D. Tseng, and M. Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 429–446, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Y. Lindell. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, chapter How to Simulate It – A Tutorial on the Simulation Proof Technique, pages 277–346. Springer International Publishing, Cham, 2017.

G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Software - Concepts and Tools*, 17(3):93–102, 1996.

J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, and S. Guillou. How to explain zero-knowledge protocols to your children. In G. Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 628–631, New York, NY, 1990. Springer New York.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

B. E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*, volume 203 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2nd edition, 2001.

A. Schmidt. Quantum algorithm for solving the discrete logarithm problem in the class group of an imaginary quadratic field and security comparison of current cryptosystems at the beginning of quantum computer age. In G. Müller, editor, *Emerging Trends in Information and Communication Security*, pages 481–493, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

P. Schulz and C. Schaffner. Lecture notes in basic probability and statistics, 2018.

M. Sipser. *Introduction to the Theory of Computation*. Course Technology, third edition, 2013.

H. van Ditmarsch and B. Kooi. *Semantic results for ontic and epistemic change*, pages 87 – 117. Texts in Logic and Games 3. Amsterdam University Press, 2008.

H. van Ditmarsch and L. B. Kuijer. Knowledge without complete certainty. In R. Iemhoff, M. Moortgat, and R. J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, volume 11541 of *Lecture Notes in Computer Science*, pages 619–632. Springer, 2019.

H. van Ditmarsch, J. van Eijck, I. Hernández-Antón, F. Sietsma, S. Simon, and F. Soler-Toscano. Modelling cryptographic keys in dynamic epistemic logic with DEMO. In J. B. Pérez, M. A. Sánchez, P. Mathieu, J. M. C. Rodríguez, E. Adam, A. Ortega, M. N. M. García, E. Navarro, B. Hirsch, H. L. Cardoso, and V. Julián, editors, *Highlights on Practical Applications of Agents and Multi-Agent Systems - 10th International Conference on Practical Applications of Agents and Multi-Agent Systems*, volume 156 of *Advances in Intelligent and Soft Computing*, pages 155–162. Springer, 2012.

H. van Ditmarsch, A. Herzig, E. Lorini, and F. Schwarzentruber. Listen to me! public announcements to agents that pay attention — or not. In D. Grossi, O. Roy, and H. Huang, editors, *Logic, Rationality, and Interaction*, pages 96–109, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

H. van Ditmarsch, W. van der Hoek, J. Halpern, and B. Kooi, editors. *Handbook of Epistemic Logic*. College Publications, 2015.

J. van Eijck and M. Gattinger. Elements of epistemic crypto logic. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, page 1795–1796, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

J. van Eijck and S. Orzan. Epistemic verification of anonymity. *Electron. Notes Theor. Comput. Sci.*, 168:159–174, 2007.

J. van Eijck and F. Schwarzentruber. Epistemic probability logic simplified. *Advances in Modal Logic*, 10, 2014.

Y. Wang. *Epistemic Modelling and Protocol Dynamics*. PhD thesis, Universiteit van Amsterdam, 2010.

R. J. Wilson. *Introduction to Graph Theory*. Prentice Hall/Pearson, New York, 2010.

W. Yu. Constructing atomic commit protocols using knowledge. NIK '04, 2004.