Formal and Computational Cryptography: Protocols, Hashes and Commitments

Flavio Garcia

ISBN: 978-90-9023009-2
IPA dissertation series: 2008-14

This thesis is typeset using LaTeX.

Cover: Zimmermann Telegram, written by German Foreign Secretary Arthur Zimmermann, is an encrypted message sent to Mexico, proposing a military alliance against the United States. It was intercepted and decrypted by British cryptographers. The telegram inflamed American public opinion and draw the United States to declare war against Germany in 1917. The bookmarker contains the telegram as partially decrypted by the British cryptographers of Room 40.

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).

Formal and Computational Cryptography: Protocols, Hashes and Commitments

A scientific essay in Natural Science, Mathematics
and Computer Science.

Doctoral thesis

to obtain the degree of doctor
from Radboud University Nijmegen
on the authority of the Rector Magnificus, Prof. dr. S.C.J.J. Kortmann,
according to the decision of the Council of Deans
to be defended in public on Friday, 9th May 2008
at 10:30 hours

by

Flavio Dario Garcia

born in Buenos Aires, Argentina
on 25 October 1978.

Supervisor:

prof. dr. Bart P.F. Jacobs

Cosupervisor:

dr. Jaap-Henk Hoepman

Doctoral Thesis Committee:

prof. dr. Tanja Lange	Technical University Eindhoven, The Netherlands.
prof. dr. Sjouke Mauw	University of Luxembourg, Luxembourg.
prof. dr. Eric R. Verheul	

Formal and Computational Cryptography: Protocols, Hashes and Commitments

Een wetenschappelijke proeve op het gebied van de
Natuurwetenschappen, Wiskunde en Informatica.

Proefschrift

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus, prof. mr. S.C.J.J. Kortmann,
volgens besluit van het College van Decanen
in het openbaar te verdedigen op vrijdag 9 mei 2008
om 10:30 uur precies

door

Flavio Dario Garcia

geboren op 25 October 1978,
te Buenos Aires, Argentinië.

Promotor:

prof. dr. Bart P.F. Jacobs

Copromotor:

dr. Jaap-Henk Hoepman

Manuscriptcommissie:

prof. dr. Tanja Lange	Technical University Eindhoven, The Netherlands.
prof. dr. Sjouke Mauw	University of Luxembourg, Luxembourg.
prof. dr. Eric R. Verheul	

Summary

In modern society we are surrounded by distributed systems. Most electronic devices that are currently on the market have some networking capability or are able to communicate with each other. Communication over shared media is inherently insecure. Therefore, proper design of security protocols is of primary concern. The design and analysis of security protocols is a challenging task. Several protocols have been proposed in the literature which later were found to be flawed. This is a consequence of the intrinsic complexity associated with the presence of a malicious adversary.

The traditional complexity-theoretical adversarial model is realistic but complex. As a consequence of this, designing and analyzing protocols in this model is error prone. The Dolev-Yao model refers to the attacker model in which an adversary has complete control over the communication media. In this model, the adversary is not bounded in running time but is completely unable to break any cryptographic primitive. This model is satisfactory as it provides a good level of abstraction. Proofs are simpler than the complexity-theoretical ones, and therefore less error prone, still capturing most common mistakes in the design of security protocols.

This thesis addresses the problem of secure protocol design from both formal and computational perspectives and also studies the relation among them. We present four original contributions:

- We present a decentralized digital currency for peer-to-peer and grid applications that is able to detect double-spending of the coins and other types of fraud.
- We develop a formal framework for the analysis of anonymizing protocols in terms of epistemic logic. We illustrate our approach by proving sender anonymity and unlinkability of some well-known anonymizing protocols.
- We relate the Dolev-Yao model, extended with hash functions, with a realistic computational model. We use a special randomized construction to interpret hashes. We show that this model is sound and complete in presence of passive adversaries. We also show that this

model is not sound in presence of active adversaries.

- We further explore the relation between these two models considering commitment schemes and active adversaries. We propose a new stronger security notion for commitment schemes and give a novel construction that is provably secure under this definition. We illustrate the usefulness of this new machinery by giving a sound interpretation of symbolic commitments in the standard model.

Acknowledgements

This work would not have been possible without the help of many people to whom I am very grateful.

First, I would like to thank my supervisor, Jaap-Henk Hoepman, who brought me from 'the other side of the world' and who believed that I was able to do a PhD, while I was not even sure myself. I also thank him for his friendship and the enjoyable 'mate' meetings we had while he was teaching me how to write papers. Next to him, I want to thank my promotor Bart Jacobs for the freedom they gave me to do research and the very relaxed and fun atmosphere they generated in the group. I also thank them for (the painful task of) proofreading this manuscript and providing insightful comments.

I especially thank Peter van Rossum for the great joy I had working with him and for everything he taught me. Furthermore, I thank him for translating the summary of this thesis to Dutch and so many other everyday things he helped me with.

I am grateful to the members of the reading committee Tanja Lange, Sjouke Mauw, and Eric Verheul for assessing my thesis.

Thanks are due to Ricardo Corin and David Galindo for their constant advise and for providing comments on early drafts of this manuscript.

Special thanks to my dear paranymphs and friends Igor and Miguel for their steadfast support and all the great fun we had together with David, the countless parties, trips, dinners and everything.

I also want to thank my friend Klasien who helped me with the design of the cover of this thesis.

Along the way I had the luck of making new friends on this side of the of the puddle Cees, Dani, Dirk, Patricia and Vero that together with my friends from the other side Alfre, Facu, Laly, Laurita, Naty, Pedro, Seba and Silvi I had very enjoyable time.

It was a pleasure (and still is) to work with my colleagues from the SoS Group. I explicitly want to thank Alejandro, Ana, Chris, Christian, Desiree, Engelbert, Erik, Harco, Ichiro, Joe, Jr., Łukasz, Maria, Mo, Olha, Pieter, Ronny and Wojciech for being such a nice fellows.

I thank my undergraduate professors from FaMAF Fransisco Tamarit (Pancho), Diego Vaggione and Javier Blanco for their encouragement and for preparing me to do research.

Finalmente, quiero agradecer a mis padres Silvia y Carlos, que a pesar de la distancia los tengo siempre presentes y que sin su apoyo incondicional nunca podría haber llegado a terminar mi doctorado ¡gracias! ...y perdón por todas las lagrimas que la distancia nos ha causado y por el tiempo que no pasamos juntos.

Flavio Garcia
Nijmegen, March 2008.

Contents

Summary **vii**

Acknowledgements **ix**

1 Introduction **1**
1.1 Cryptographic Primitives 2
1.2 Security Goals 7
1.3 Design and Analysis of Security Protocols 8
1.3.1 The System and Adversarial Models 8
1.3.2 Challenges in Design and Analysis 10
1.3.3 Relating these two Adversarial Models 10
1.4 Overview of this Thesis 11

2 Protocol Design using Hashes for E-currency **15**
2.1 Introduction 15
2.1.1 Related Work 16
2.1.2 Our Contribution 17
2.1.3 Structure of the Chapter 18
2.2 System Model 19
2.2.1 Notation 20
2.3 System Objectives and Threat Model 21
2.3.1 Threat Model 21
2.3.2 System Objectives 21

2.4 The Off-Line Karma Protocol 22
2.4.1 Informal Description 22
2.4.2 Off-Line Karma for Static Networks 23
2.4.3 Handling Dynamic Networks 31
2.5 Early Double-spending Detection 34
2.6 Conclusions 35

3 Protocol Analysis of Anonymity in the Formal Setting 37
3.1 Introduction 37
3.2 The message algebra 40
3.3 Runs and observational equivalence 43
3.3.1 Network model 43
3.3.2 Attacker model 43
3.3.3 Communication protocol 44
3.3.4 Observational equivalence 45
3.4 Formulas and epistemic operators 48
3.4.1 Formulas 48
3.4.2 Epistemic operators 50
3.4.3 Expressing information hiding properties 50
3.5 Examples 52
3.5.1 Crowds 52
3.5.2 Onion Routing 53
3.5.3 Onion Routing with subtle flaw 57
3.6 Conclusions and Future Work 59

4 From Formal to Computational Proofs of Protocols using Hashes 61
4.1 Introduction 61
4.2 The Symbolic Setting 64
4.3 The Computational Setting 68
4.3.1 Probabilistic Algorithms 69
4.3.2 Encryption Scheme 70
4.3.3 Oracle Hashing 72
4.4 Interpretation 74

4.4.1 Partial interpretation 76
4.5 Soundness 78
4.6 Completeness 87
4.7 Active adversaries 97
4.8 Conclusions 99

5 Extending Computational Soundness Further: the case of Non-malleable Commitments 101
5.1 Introduction 101
5.2 Symbolic Protocols 103
5.3 Computational Setup 105
5.3.1 Commitment Schemes 105
5.3.2 Encryption Schemes 106
5.3.3 One-time Signatures 107
5.3.4 Tag-based Encryption 108
5.3.5 Interpretation 109
5.4 Definitions of Non-malleability 109
5.4.1 NMC-CCA: Non-malleability Against Chosen Commitment Attacks 111
5.4.2 An Indistinguishability Based Definition 112
5.5 The Construction 113
5.6 Protocol Execution and State Traces 117
5.7 Conclusions 121

Bibliography 123

Samenvatting (Dutch Summary) 139

Curriculum Vitae 141

Chapter 1

Introduction

Originally, computers where isolated (rather big) pieces of machinery designed to accomplish some task. With the widespread use of computing systems there was an increasing need for intercommunication. Computer networks emerged and with them the need to establish conventions for communication, since these computers potentially have different architectures and operating systems. A *protocol* (coming from the Greek protocollon, which was a leaf of paper glued to a manuscript volume, describing its contents) involves two or more parties, and describes a sequence of steps that each party must perform. This sequence of steps must be precisely defined and should be executed in order. We deal with protocols in everyday life. A simple example of a communication protocol is the one of the walky-talky. A walky-talky is a bi-directional radio transceiver where only one radio transmits at a time. In this protocol each party must push a button while talking and finish the message with the keyword 'over'. This indicates to the other party that it is the end of the message and that the communication channel becomes available for transmission again. Obviously, if one party deviates from the protocol and omits the 'over' keyword, then the other parties would wait indefinitely. The Internet Protocol (IP) is probably the most widely used network protocol at present. Other protocols like POP3 and HTTP are also used in the 'everyday life' and greatly contributed to

the expansion and success of the Internet.

As networks expand, say outside a company or institution, the network and parties become untrusted. Then, we need special protocols that can achieve some properties like authentication and secrecy (defined later in Section 1.2) even when some malicious entity is in control of the network and/or some of the parties.

A *security* or *cryptographic protocol* is a protocol that has a *security goal*. Security goals are properties that try to withstand malicious attempts to abuse the system. In a protocol, each party is called *principal* or *agent* and might be a human, computer, smart card, etc. It is customary to name these principals Alice, Bob, Charlie and so on. Principals do not necessarily trust each other and the communication media is also, in general, not trusted. It is assumed to be reliable in the sense that transmission errors are absent. Principals execute the protocol by exchanging a sequence of messages. The protocol is *secure* if the security goal is fulfilled upon termination. An *attacker* is a principal that does not necessarily follows the protocol. In fact, her main objective is to break the security goal. The attacker is often called Trudy or Eve. Security protocols are the subject of study of this thesis.

Now one might wonder how it is possible to achieve any security in such an adverse scenario. This is where cryptography comes into play.

1.1 Cryptographic Primitives

Cryptographic primitives are the basic building blocks for security protocols. *Cryptography* is the art (or science) of secret writing and was first and briefly used by the Egyptians, 4000 years ago. A simple substitution cipher, so-called Atbash cipher, was used by the Hebrews (around 600 BC). This cipher consisted of a reversed ordered alphabet. In order to encrypt a message, every occurrence of the letter A was replaced by Z, B by Y and so on. The Spartans (around 400 BC)

Spartan Skytale

used a belt, rolled on a random-diameter cylinder, called Skytale to write their secret messages. The diameter of the cylinder was a shared secret between the originator and receiver of the message. Having just the belt with some letters written on it, it is hard to recover the original message. Julius Caesar (100 BC) also used a substitution cipher consisting of shifting each letter of the alphabet by a fixed (secret) amount. All these are examples of *symmetric key cryptosystems*.

Plaintext:	A	T	T	A	C	K	G	A	U	L
Ciphertext:	D	W	W	D	F	N	J	D	X	O

Figure 1.1: Caesar's cipher with a shift of three letters

Their purpose is to facilitate secure and private communication among principals, over an insecure channel. It is assumed that these principals share a secret, called the (secret) *key*. The message, called *plaintext*, must remain secret to anyone that is not in possession of the secret key. To achieve this, the plaintext is encoded and scrambled with the key in such a way that an observer cannot extract any useful information from it. This encoded message is called *ciphertext*. On the other hand, a principal that is in possession of the secret key should still be able to recover the original message.

Modern encryption schemes achieve much stronger security properties than their historical counterparts. Probably the simplest encryption scheme known is also the best: the *one-time pad*, invented by Mauborgne and Vernam [Kah67] in 1917. The key consists of a sequence of random bits of the same length as the plaintext. In order to encrypt a message, just compute an exclusive or operation (XOR) with the key. In order to decrypt just XOR the ciphertext with the symmetric key, recovering the original plaintext. This simple encryption scheme is perfect in the sense that it perfectly hides any information about the plaintext. There are few practical inconveniences for its use such as the fact that sender and receiver must share a key of the same length as the message, and that this key cannot be reused.

The Data Encryption Standard (DES), has been a worldwide standard for decades. It is a block cypher where the message and the key are scrambled together by a sequence of permutations, substitutions and XOR operations. The design principles of DES have not been revealed but it resisted years of cryptanalysis. However, due to short key length, it was vulnerable by a hardware implementation of a *brute force attack*. This attack simply consists of trying exhaustively every possible key. Its successor, Advanced Encryption Standard (AES) has nicer mathematical properties and significantly longer key sizes. So far there are no known attacks to the full version of it and is considered the state of the art in symmetric key cyphers.

A revolutionary new concept, *public key cryptography* was introduced by Diffie and Hellman in [DH76]. Concurrently and independently, this notion was also proposed by Ralph Merkle in [Mer78] but, due to a slow publishing process, his contribution was made public almost two years later. The revolutionary idea behind public key cryptography is that encryption and decryption keys could be different. Therefore, depending on the application, one of them could be made public. This lead to a number of new applications like digital signatures [RSA78b] and key agreement [DH76], among others. In contrast with symmetric key cryptography, with public key cryptography it is suddenly possible, for two principals that do not share any secret, to establish a private channel. The first practical implementation of a public key encryption scheme, was published by Rivest, Shamir and Adleman [RSA78a].

Digital signatures are the digital counterpart of traditional signatures. The signer has a secret signing key and the corresponding verification key is made public. The binding between the public key of the signer and his identity must be certified by the so called *certification authority*. A certification authority is a trusted party, that after authenticating a person and having some evidence of the possession of the private key, issues a *public key certificate*. This public key certificate is a statement where the certification authority certifies that the claimed public key actually belongs to his identity. This certificate is signed by the certification authority itself. In order to verify that a public key certificate is valid, principals need to have access to the public key of the certification authority. This is

considered to be well known, and in practice it is incorporated into web-browsers and other network clients.

Cryptographic hash functions (hereafter, simply hash functions), play a fundamental role in modern cryptography. They are basic building blocks for signature schemes [Dam88], encryption schemes [CS03b], message authentication codes [Tsu92] and they are also widely used as fingerprint for any kind of digital data. A hash function h maps bitstrings of arbitrary finite length to bitstrings of fixed length. It should be efficient to compute and hard to invert. This means that given a bitstring x it is easy to compute the hash value $h(x)$, but given $h(x)$ it is infeasible to compute x. Moreover, it is infeasible to find any bitstring y such that $h(y) = h(x)$. Hash functions are many-to-one, therefore there must be different values x and y for which $h(x) = h(y)$. Such values are called *collisions*. Another security requirement for hash functions, *collision resistance* (aka. strongly collision resistance) is that it is infeasible to find any collision.

Commitment schemes are also one of the fundamental cryptographic primitives and are used in protocols like zero-knowledge proofs [GMW91], secure multiparty computation [GMW87], contract signing [EGL85], and can be directly used for bidding protocols. A commitment consists of two phases: the *commitment phase* where the principals commit to a message without revealing any information about it; and the *opening phase* where the principals reveal the message and it is possible to verify that this message corresponds to the committed value in the commitment phase. After the commitment phase it should be infeasible to open the commitment to a different value than the one committed. This property is called *binding*.

Semantic security is an intuitive and by now standard security notion. It was proposed by Goldwasser and Micali in their seminal paper Probabilistic Encryption [GM84a], where they also establish the foundations of modern cryptography. Intuitively, semantic security means that it is infeasible for the adversary to 'learn anything' about the plaintext by looking at the ciphertext. This is important in practice as when keys are being encrypted, leaking a single bit reduces de key search time by a factor of two. This is an appropriate security notion while considering eavesdropping adversaries.

While considering an adversary that is able to intercept and modify

messages, semantic security is not strong enough. In particular it does not guarantee that the adversary is not able to modify a ciphertext into another ciphertext that decrypts to a related plaintext, even without knowing the original plaintext. This notion is known as *malleability* and was introduced by Dolev, Dwork and Naor [DDN91]. The notion of malleability also extends to other cryptographic primitives like signature and commitment schemes. Malleability is a real problem in practice; imagine a situation where we are bidding for a certain good, and we are using a malleable commitment scheme to do so. It is clear that if the adversary could modify our bid, let us say, by adding one to it, then we would not be able to win the auction. Non-malleability is hard to achieve and it is usually overlooked by practitioners while designing security protocols. Traditionally it was assumed that non-malleability somehow followed from the fact that the set of valid plaintexts was sparse in the set of all possible bitstrings. In fact, the first practical non-malleable encryption scheme under 'realistic assumptions' was due to Cramer and Shoup [CS03a] in 1998, almost ten years after non-malleability was proposed. For the case of hashes, there are still no known constructions for non-malleable hashes that have been proven secure. Even worse, an equivalent notion to non-malleability for hashes has not been formulated. Particularly for hashes, this constitutes a gap between the formal analysis of security protocols and the underlying cryptographic assumptions, as in the former, non-malleability for hashes is implicitly assumed.

So far, we have intentionally been using the word *infeasible* without a precise meaning. This is because its meaning depends on the adversarial model. This notion will be defined later in Sections 1.3.1 and 1.3.1.

It is hard to describe in a realistic manner what the capabilities of the attacker are. As a matter of fact, the adversarial model seems to be evolving over the years: as new attacks are found, the adversarial model be comes stronger.

1.2 Security Goals

Next, we briefly describe some of the most common security goals. This list is, of course, incomplete and vague as security goals in practice vary as much as the problems that people intend to solve.

1. *Confidentiality* has always had a central role in cryptography and it has a clear meaning: some sensitive message remains unknown to everyone except its intended recipients.
2. *Authenticity* is when the recipient of a message is certain of its originator. When initiating a communication it also means that initiator and responder know the identity of each other (mutual authentication). This property is harder to verify than secrecy and there is a long history of flawed protocols that intended to achieve it.
3. *(Data) integrity* is when the receiver of a message is able to check whether the message has been modified during transmission or not.
4. *Availability* means that a certain data (or service) is reachable (or working properly) when needed.
5. *Non-repudiation* means that the sender of a message is not able to deny, later, that he sent this message because there is a non-refutable proof that he did it.
6. *Anonymity* is the property of remaining unknown. Such a property is highly desired in some contexts like internet voting. An *anonymizing protocol* is a protocol that has the goal of hiding the identities of the users.
7. *Unlinkability* is another information hiding property and it is slightly more subtle than anonymity. Intuitively, it means that the adversary does not know if two principals are communicating with each other at all.
8. *Plausible deniability* is the capability of a principal of proving that he does not possess a certain knowledge.

1.3 Design and Analysis of Security Protocols

In order to design and analyze a security protocol we first need to define in which environment this protocol is supposed to run. To do that we need to define the capabilities and limitations of the principals (including the adversary) and the communication media. Then we need to define what kind of properties this protocol is supposed to fulfill.

1.3.1 The System and Adversarial Models

The system model includes *honest participants* that behave exactly as specified in the protocol. We assume that each of these participants has a secure environment in which to compute, such as a personal computer or smart card. These participants perform actions like communication and computations. Depending on the network model the communication might be asynchronous, private, anonymous, authenticated, etc. In order to be able to determine the security of a protocol we need to define what are the capabilities of the adversary. Next we describe some common adversarial models.

The Dolev-Yao Model

The Dolev-Yao (aka. symbolic or *algebraic*) model refers to the attacker model proposed by Dolev and Yao in [DY83], following the directions of Needham and Schroeder [NS78]. In this model, the adversary has complete control over the communication media. Principals deliver every message to the adversary and she might forward, destroy, delay or modify each message at will. Moreover, she is capable of synthesizing new messages and deliver these new messages to honest participants. There are well defined rules constraining what kind of messages the adversary is allowed to build. Intuitively, these are the messages that she can build by combining the information she learns from previously seen messages, her private information and some fresh randomness. In this model, the adversary is not bounded in running time but is completely unable to break any cryptographic primitive. For instance, this means that if the adversary sees a ciphertext for

which she does not know the decryption key, then she learns nothing about the original plaintext. Similarly for hashes, a hash value $h(x)$ does not reveal any information about the pre-image x. This extends to any other security properties of the cryptographic primitives, like non-malleability and collision resistant. This constitutes a fairly strong assumption.

The main advantage of this model is that it provides a good level of abstraction. Proofs are much simpler and therefore less error prone than the computational ones, still capturing most common mistakes while designing security protocols. This follows the principle that "Cryptography is not broken, it is circumvented" as Shamir said. A key advantage of this model is that there are tools like ProVerif [Bla01] and Scyther [Cre06] that are capable of automatically proving some security goals such as authentication and secrecy.

Traditional Computationally-Bound Adversary

In this model, principals are probabilistic polynomial-time Turing machines and so is the adversary. The *security parameter* is a natural number in unary representation. These Turing machines take as an argument the security parameter and are allowed to run in polynomial time on the size of their input. The intuition is that breaking a cryptographic primitive requires super-polynomial time. Therefore, by increasing the security parameter, the protocol becomes arbitrarily hard to break. In this context, algorithms define probability distributions over bitstrings and security is usually defined in terms of computational indistinguishability of these distributions. What is feasible for an adversary in this model is simple: any polynomial-time computation.

The main advantage of this model is also its main disadvantage: it considers every possible behavior for an attacker. As a consequence proofs are very complicated and error prone. But how is it at all possible to show that there is no efficient algorithm that can break the security goal? Here is where the *computational assumptions* come in to play. There is a (small) number of problems that are considered to be of super-polynomial complexity (e.g., factoring RSA integers or taking logarithms in cryptographically

strong groups), mainly because the scientific community has spent 'enough' effort trying to find efficient solutions to these problems and failed. After some time these assumptions became widely accepted. Then, the proof strategy for a security protocol consists in reducing the security of the protocol to one of these computationally hard problems. The argument runs as follows: if the protocol is broken then we solve a hard problem, but this is (conjectured to be) not possible, ergo the protocol is secure.

1.3.2 Challenges in Design and Analysis

The design and analysis of security protocols is a challenging task. Several protocols have been proposed in the literature e.g., [NS78, DS81] which many years later were found to be flawed [Low95, Low96, Low97]. This is not due to negligent design but it is a consequence of the intrinsic complexity associated with the presence of a malicious adversary. This makes it difficult to use standard debugging techniques such as model checking. Beyond standard security notions like authentication and secrecy there are no well established techniques for the analysis and design of security protocols, for example, for digital currencies or anonymity. Another problem is the lack of modularity: in general, security protocols are not compositional. This means that having two protocols that are secure does not imply that these protocols together are secure. Canetti, in his universal composability framework [Can01], took the first steps addressing this problem. In general, non-malleable cryptographic primitives are of great aid towards modular design of security protocols.

1.3.3 Relating these two Adversarial Models

Abadi and Rogaway in [AR02] pioneered on the idea of relating these two adversarial models and showed that, under appropriated assumptions on the underlying cryptographic primitives, a simple language of encrypted expressions is sound with respect to the computational model, while considering passive adversaries. Much more work has been done in this direction [MW04b, MP05, Her05, GvR06b] considering different cryptographic

primitives and adversarial models.

The relation between these two models consists of a map form symbolic messages m, to distributions over bitstrings $[\![m]\!]$. This map should relate messages that are observationally equivalent in the symbolic world to indistinguishable distributions over bitstrings. Such a map allows one to use formal methods, possibly even automated, to reason about security properties of protocols and have these reasonings to be valid also in the computational world.

This technique is useful as it combines the advantages of each model. On the one hand proofs are simple, less error prone and can be verified by formal analysis techniques. On the other hand, these proofs have associated a proof in the computational setting considering all possible probabilistic polynomial time adversaries and standard assumptions on the cryptographic primitives.

So far there is no work in the literature that we are aware of that relates this models for hash functions and commitment schemes, which would greatly aid in formal analysis techniques.

1.4 Overview of this Thesis

The contribution of this thesis is on protocol design and analysis addressing the challenges described in Section 1.3.2 supported by computational soundness techniques addressing the issues exposed in Section 1.3.3. In this thesis we propose four original contributions which we summarize below. Unless otherwise mentioned, each chapter is based on previously published work. Below we also credit the co-authors.

Chapter 2. Protocol Design using Hashes for E-currency

This chapter describes a non-traditional application of hash functions as a building block for a digital purse for decentralized Peer-to-peer systems. Peer-to-peer (P2P) and grid systems allow their users to exchange information and share resources, with little centralized or hierarchical control,

instead relying on the fairness of the users to make roughly as much resources available as they use. To enforce this balance, some kind of currency or barter (called *karma*) is needed that must be exchanged for resources thus limiting abuse. Previous proposals for such systems have not taken the decentralized and non-hierarchical nature of P2P and grid systems into account. The system is based on tracing the spending pattern of coins, and distributing the normally central role of a bank over a predetermined, but random, selection of nodes.

This chapter is based on the article "Off-line Karma: A Decentralized Currency for Peer-to-peer and Grid Applications" [GH05a] by Jaap-Henk Hoepman and the author, which subsumes the article "Off-line Karma: A Decentralized Currency for Static Peer-to-peer and Grid Networks" [GH05b] by the same authors.

Chapter 3. Protocol Analysis of Anonymity in the Formal Setting

This chapter provides a formal framework for the analysis of information hiding properties of anonymous communication protocols in terms of epistemic logic. The key ingredient is the notion of observational equivalence, which is based on the cryptographic structure of messages and relations between otherwise random looking messages. Two runs are considered observationally equivalent if a spy cannot discover any meaningful distinction between them. We illustrate our approach by proving sender anonymity and unlinkability for two anonymizing protocols, Onion Routing and Crowds. Moreover, we consider a version of Onion Routing in which we inject a subtle error and show how our framework is capable of capturing this flaw.

This chapter is based on the article "Provable Anonymity" [GHPvR05] by Ichiro Hasuo, Wolter Pieters, Peter van Rossum and the author.

Chapter 4. From Formal to Computational Proofs of Protocols using Hashes

This chapter provides one more step towards bridging the gap between the formal and computational approaches to the verification of cryptographic

protocols. We extend the well-known Abadi-Rogaway logic with probabilistic hashes and give a precise semantic interpretation of it using Canetti's oracle hashes. These are probabilistic polynomial-time hashes that hide all partial information. We show that, under appropriate conditions on the encryption scheme, this interpretation is computationally sound and complete. This can be used to port security results from the formal world to the computational world when considering passive adversaries. We also give an explicit example showing that oracle hashing is not strong enough to obtain such a result for active adversaries.

This chapter is based on the articles "Sound Computational Interpretation of Symbolic Hashes in the Standard Model" [GvR06b] by Peter van Rossum and the author and the unpublished "Completeness of Formal Hashes in the Standard Model" [GvR06a] available as preprint, by the same authors. A journal version, consisting of a compound of this two articles, appears as [GvR08].

Chapter 5. Extending Computational Soundness Further: the case of Non-malleable Commitments

This chapter aims to find a proper security notion for commitment schemes to give a sound computational interpretation of symbolic commitments. We introduce an indistinguishability based security definition of commitment schemes that is equivalent to non-malleability with respect to commitment. Then, we give a construction using tag-based encryption and one-time signatures that is provably secure assuming the existence of trapdoor permutations. Finally, we apply this new machinery to give a sound interpretation of symbolic commitments in the Dolev-Yao model while considering active adversaries.

This chapter is based on the article "Computational Soundness of Non-Malleable Commitments" [GGvR08] by David Galindo, Peter van Rossum and the author.

Chapter 2

Protocol Design using Hashes for E-currency

2.1 Introduction

Peer-to-peer (aka. *P2P*) networks like the widely used BitTorrent [Coh03] and Freenet [CSWH01], and grid systems [BHF03] are distributed systems without centralized control or hierarchical organization. Given this flat structure, these systems scale very well when the number of nodes increases. Scalability is important, given the fact that the Internet is still growing exponentially and more and more people have permanent Internet access with flat rates. Current applications of these systems include but are not limited to: file sharing, redundant storage, distributed computations, data perpetuation, P2P e-mail, P2P web cache and anonymizing systems.

Grid systems capitalize on the observation that computer resources are usually very badly distributed in both time and space, and often wasted. CPU cycles are maybe the best example of this. Most of the computers in the world are idle most of the time, with only occasional periods of high load. Then, it seems natural to make resources available when idle, and to be able to use the resources of other users in return, when needed. In an ideal grid system, the whole Internet constitutes a huge supercomputer

with practically unlimited resources, that members can use as long as they contribute to it as well. Projects like seti@home, folding@home and distributed.net have shown that a large set of common desktop computers can provide a tremendous amount of computing power. Even though they receive no direct benefit, users participate in such projects because they associate themselves with the goals of the project. If such large scale computations are for an uncompelling cause, it is not easy to find people willing to donate their CPU time.

Also, many P2P networks suffer from the 'free-riders' problem where users only occasionally connect to the network to use the resources offered by it, but do not donate any resources themselves. Adar and Huberman [AH00] performed a traffic study on the Gnutella network revealing that 70% of the users share *no* files at all. To counter such problems, 'currencies' of some sort have been proposed to reward users contributing to the system and that can be used as payment when the resources of the network are used.

2.1.1 Related Work

Several P2P systems like POPCORN [NLRC98] and MojoNation[1] use some kind of digital currency to enforce contribution and optimize resource distribution. All these systems use a central bank or broker to track each user's balance and transactions. Micropayment schemes [GMA+95, RS97, Riv04, Pár05] like Rivest and Shamir's PayWorld and MicroMint [RS97], Glassman et al's Millicent [GMA+95] and Rivest's Pepercoin [Riv04] seem to be especially suitable for such a task. However, these schemes are centralized and the load of the central broker grows linearly with the number of transactions. It is clear that when scalability is of primary concern, a central bank or broker constitutes both a bottleneck as well as a single point of failure. In the context of P2P networks, avoiding lawsuits is an extra motivation for having a decentralized system, i.e., there is no single entity that can be sued but just a large number of regular users of the system.

[1] MojoNation no longer exists, but the website has been archived. See `web.archive.org/web/*/mojonation.net/*`.

The aforementioned approaches are therefore not suitable to fully solve our problem. Furthermore, there is no trivial way to decentralize them given that the trusted bank plays an important role in such a scheme.

At the moment, the only distributed currency we are aware of that is fully decentralized is KARMA [VCS03]. In that system, the bank for a user is distributed over a bank set of r users, that all need to be on-line, and that are all involved in every transaction between their "owners". KARMA splits the bank functionality in different bank sets, which are sets of users of size r. Each of these bank sets is responsible for keeping the state balance of a set of users. In KARMA, every transaction between users Alice and Bob involves communication between them, between Alice and Bob and their bank sets, and most importantly, it involves r to r communications between the bank set of Alice and the bank set of Bob. This incurs a large overhead, especially in cases where the transaction rate is high.

Another interesting approach is PPay [YGM03]. PPay is a lightweight micropayment scheme for P2P systems. In PPay the issuer of the coin is responsible for keeping track of it. With every transaction the issuer of the coin updates a pointer to the new owner, in a secure manner. The main drawback with PPay is that it uses a central server (called broker) when the issuer of a coin is off-line. This means that when Alice, who owns a coin minted by Carol who is off-line, wants to spend it at Bob, Alice should make the transaction via a central broker. In some frameworks, were the ratio of off-line users is high or in very dynamic systems were users join at some point and never reconnect again, the probability of finding the original issuer of the coin on-line is very low. Therefore, in certain situations PPay converges to a system with a centralized accounting bank.

2.1.2 Our Contribution

We present a completely decentralized, off-line karma implementation for *dynamic* P2P and grid systems, that detects double-spending and other types of fraud under various adversarial scenarios. The system is based on the tracing of the spending pattern of coins, and distributing the normally central role of a bank over a predetermined, but random, selection of nodes.

Transactions between users do not require the cooperation of this distributed bank — this is more efficient, but as a result double spending cannot be prevented. Instead, karma coins need to be occasionally reminted to detect fraud. The system is designed to allow nodes to join and leave the system at arbitrary times. In our system, a transaction only involves communication between the two users exchanging the coin. As a trade off, we can only provide double-spending detection instead of prevention. For an alternative approach we refer the reader to [Hoe07].

We focus on the payment for CPU cycles as an application of our techniques, and show how a special minting technique allows us to initialize the system and provide its users with coins in a quite autonomous fashion. Coins correspond to CPU cycles, and the bag of coins owned by a user corresponds, in a sense, to a battery charged with CPU cycles. The battery is initially empty, and can be charged by minting coins. Minting takes quite a few CPU cycles. Alternatively, a coin can be obtained by performing roughly the same amount of work, but for another user. Extensions of our protocols to trade coins for other resources are certainly possible, and only involves initializing the system with a set of coins in a different manner.

In order to provide some abstraction, we build our system on top of an arbitrary overlay network which provides certain services as described in Section 2.2. The reader can think in a layered structure where TCP/IP is at the bottom, above it there is a P2P routing layer, followed by a secure routing layer, and the off-line karma protocol on top. We stress that the off-line karma protocol we present here could be implemented on an arbitrary P2P network that supports secure routing, or on which a secure routing layer can be implemented.

2.1.3 Structure of the Chapter

The remainder of this chapter is organized as follows. Section 2.2 discusses the system model and notation used throughout the chapter. We describe the system objectives and the capabilities of the adversary in Section 2.3. Then we present our karma implementation in Section 2.4, first for a static network and then the dynamic case. Finally, Section 2.5 discusses methods

for early double-spending detection, and Section 2.6 presents our conclusions and directions for further research.

2.2 System Model

In the context of this work we want to stay abstracted from the underlying overlay network. We are going to model common characteristics that apply to routing overlays like CAN [RFH+01], Chord [SMLN+03], Pastry [RD01] and Tapestry [ZHR+04] as in [CDG+02], where the reader can find also a nice and brief description of each system. In this abstract model, every node that joins the system is assigned a uniform random identifier u from the identifier space Π. We assume that the overlay network provides primitives for both user look-up and message routing. Furthermore, for each possible identifier u (whether u is part of the network or not) the overlay network can efficiently and reliably compute the *neighbour set* $\aleph_r(u)$, which consists of r on-line nodes *close* to u. The definition of *close* varies in each of the above mentioned systems, although it is always well-defined. We also assume that communication within this set is *inexpensive*, because nodes keep updated routing information of their neighbors. Given that the node identifiers are distributed randomly, any neighbour set represents a random sample of all participating nodes [CDG+02].

Off-line Karma requires every user to have his own public key pair and a certificate that binds the public key with a node identifier. This may be provided by a trusted *Certification Authority* (hereafter CA). We want to remark that the CA is only needed when a new user joins the system. After that communication with the CA is no longer needed.

Routing information in the overlay network is kept updated, in practice, by node join and node leave messages and periodic queries and fingers to detect when a node suddenly disconnects. This mechanism introduces propagation and update delays. This is, in fact, a discrete approximation of an ideal situation where any modification in the network topology is instantaneously detected by the overlay network. We assume such an ideal situation, and leave responsibility for emulating this ideal functionality in

an efficient fashion to the overlay network. In other words assume that node joins and leaves are atomic operations.

We also assume that the overlay network is capable of safely distributing a blacklist of banned users. Whenever a user detects fraud and has a proof of that, he can submit it to the overlay network which makes this information available to every user. How to implement the distribution of blacklist securely is beyond the scope of this thesis. We assume that node joining the system is expensive with respect to the benefits of fraud.

2.2.1 Notation

We write $\{\, m \,\}_u$ for u's signature on message m, C_u for u's certificate, and $\texttt{validSig}(m, u, C_u)$ for the function that checks u's certificate C_u, and if valid uses the key in C_u to verify a signed message m.

We also use a multisignature scheme. A multisignature scheme [OO99, MOR01] is a signature scheme where a set R of users sign a message. A multisignature $\{\, m \,\}_R$ for a message m has the same properties as if each user in R concatenates his own traditional public key signature to m, the only difference is that a multisignature is more efficient in size and in verification time (comparable to a single signer Schnorr's signature scheme). Unlike a threshold signature scheme however, it does not provide anonymity. We define $C_R = \{C_i : i \in R\}$ and $\texttt{validSig}(m, R, C_R)$ as the function that checks all certificates in C_R and verifies the multisignature.

Security of our system is parameterized by a security parameter η. All cryptographic primitives we use satisfy the requirement that the advantage of the adversary breaking them is less than $2^{-\eta}$. We show that the advantage breaking our karma system is at most that large too.

For describing protocols we adopt the notation $a \to b : m \to m'$ to denote that Alice sends a message m to Bob which he receives as m'. Also $a : f$ means Alice computes f. If f is a predicate, Alice verifies $f \equiv \mathit{true}$ and aborts if not.

2.3 System Objectives and Threat Model

2.3.1 Threat Model

We consider a set of n users U of which at most t are under control of the adversary. In the context of P2P networks, there is an important difference in the difficulty for an adversary between adding new corrupted users to the system and getting control over chosen user identities. Therefore, we also define $0 \leq c \leq t$ to be the number of corrupt user identities chosen by the adversary after they joined the overlay network. Then, when $c = t$ we give the adversary full control over which nodes in the overlay get corrupted, while for $c = 0$ the adversary is only able to get a set of randomly chosen user identities of size t.

Furthermore, we assume that the adversary cannot make excessively many nodes join and leave the system, or let some nodes join and leave in a very high frequency (in attempts to mount sybil attacks, to use them as strawmen, or to overcome the random assignment of node identifiers). In fact, we do not allow the adversary any control over when nodes join the system. In practise, this could be achieved by requiring nodes to pay each time they register, or making node joins a time-intensive procedure (e.g., by requiring them to compute a moderately hard, memory bounded function [ABMW05, DGN02]).

2.3.2 System Objectives

We note that for any system offering off-line currency, double-spending *prevention* is generally speaking not possible, unless extra assumptions (e.g., special tamper proof hardware) are made. As we are designing an off-line system, we only require double spending *detection*. We focus on the payment itself and not on the exchange of coins for goods. We do not consider issues like fair exchange or coin stripping. These notions are the the equivalent of the real life scenario where the buyer halves a banknote and gives one half before and the other half after the goods have been delivered.

Then, the requirements on a usable, off-line and decentralized, karma system for P2P and grid applications are the following.

Scalability Transaction cost should be independent of the number of users in the network.

No centralized control The system should not rely on one or several central or special nodes such as banks or brokers and should not require any non-flat hierarchy. We do allow a centralized registration procedure.

Load Balance The overhead of the protocol must be, on average, evenly distributed over the peers.

Availability Transactions among users can be processed uninterrupted even when users join or leave the system.

Unforgeability It should be infeasible to forge coins, after an initial amount.

Double-spending detection The system must detect double-spending, and for every double spent coin, a fraudulent user must be blacklisted with overwhelming probability.

2.4 The Off-Line Karma Protocol

2.4.1 Informal Description

To implement the CPU cycles battery metaphor presented in the introduction, a user can mint coins by finding collisions on a hash function (a la hashcash [Bac97]). This rather expensive minting process is preferred over giving an initial amount of free coins to new users, as in that case the system becomes vulnerable to users changing their identities after spending those coins. A minted coin contains the name of the minting user as well as a sequence number (limiting the number of coins a single user can mint). User identity and sequence number together constitute the unique coin identity. Coins also contain a time stamp recording the time they were minted.

The coins are transferable [CP92]. A user can pay for resources by transferring a coin to another user. The sender signs the coin, thus committing

to the change of ownership. Then, the receiver verifies this signature and stores the coin (with signature) for further use. With every transfer, a coin is extended with another signature. Thus, the sequence of signatures on a coin records the payment history of that coin. Double-spending is detected by comparing the history of two coins with the same coin identity, and the culprit (or his accomplice) will be found at the node where both histories fork. This check is performed whenever a coin is reminted. Fraudulent nodes are blacklisted, together with a proof of their misbehavior (namely two signatures of the fraudulent node over the same coin). This prevents unfair blacklisting. Double-spending among corrupted nodes is *not* considered double-spending unless at some point two or more honest users get the same coin.

Every once in a while (but at least before the coin expires), coins must be reminted. Reminting is used to detect double-spending, and at the same time to reduce the size of the coin by removing its history. In classical systems, reminting is done by a central bank. Here the function of the bank is distributed over a set of users on the network called the *reminters* for the coin. The set of reminters is constructed in such a way that

- at least one of the reminters is a non-corrupted node, and
- all honest reminters possess the history of previously reminted coins with the same identity.

We first describe the static case where we assume to have a set of n users which are always on-line and later, in Section 2.4.3 we describe the modifications needed for handling dynamic networks, where users join and leave at arbitrary times.

2.4.2 Off-Line Karma for Static Networks

Minting At the beginning, users need to have an initial amount of karma; otherwise no one in the system would be able to pay for any service. An obvious solution would be to give new users an initial amount of karma, but a dishonest user could spend this karma without ever contributing to the system. A better approach is to allow users to mint their own coins.

For the minting process it is also necessary to spend CPU time in solving a cryptographic puzzle. Our proposal is to do it a la hashcash [Bac97] in the case of payment for CPU cycles. In a more general case, any proof that the minter has wasted at least as much resources as the karma coin is worth is sufficient. This puzzle is bond to the issuers identity and also to a time stamp, to avoid being reused by an adversary. Although expensive, it is still possible for a malign user to keep minting new coins forever. In this scenario the system has an increasing amount of coins, but constant potential computer power (our asset), so we are in presence of an inflationary system, which is not desirable given that each coin introduces some overload. Therefore we need a boundary for the amount of coins each user can mint otherwise the system becomes vulnerable to abuse or hyper-minting, where users just mint new coins and do not contribute providing services to the community. We prevent that by having a fixed length serial number on each coin and users are not allowed to use the same serial number twice, which would be considered double-spending.

Let $h_1 : A \to C$ and $h_2 : B \to C$ be mutually collision resistant hash functions. Note that such functions exist and can be constructed from any collision resistant function H, for instance, by adding some constant prefix to its input, i.e., $h_1 = H(c_1, x)$, $h_2(x) = H(c_2, x)$ for suitable constants c_1, c_2. Now suppose that every user is allowed to mint 2^q karma coins. A user u has to spend some CPU time finding a collision y satisfying: $h_1(x) = h_2(y)$ and $x \neq y$, with[2]

$$x = \underbrace{u||sn}_{coinId}||ts$$

where sn is the serial number $|sn| \leq q$ and ts is a time stamp. This is an expensive but feasible operation, for suitable functions h_1 and h_2. In analogy with the monetary system, imagine that the cost of the metal needed for minting a coin is greater than its nominal value. We define the new karma coin as

$$k_0 = \langle x, y \rangle$$

[2] $||$ denotes bitstring concatenation

The function id takes a karma coin and returns the coin identifier of that coin, in this example $u||sn$.

Spending To spend a coin, a user u transfers ownership of it to the merchant m, by signing the coin together with the identity of the merchant m and a random challenge z chosen by the merchant. The random challenge is included to avoid uncertainty about the culprit in the case where a user spends the same coin twice at the same merchant. Otherwise, a fair user might look like the double-spender (unless he keeps history of all received coins indefinitely). Concretely, suppose that the user s owns the coin k_i and wants to spend it at the merchant m. Then, the latter sends a random challenge z to the former, who computes:

$$k_{i+1} \quad = \quad \{k_i, z, m, C_u\}_u$$

and sends it to m. The merchant m will then verify the hash collision, the complete signature chain and make sure that his own challenge nonce and identities are included.

Reminting To prevent the coins from growing unreasonably large and to bound the amount of history that needs to be kept, coins must be reminted regularly. This procedure is enforced by letting the coins expire otherwise. Assume that there is a maximum time to live T for a coin. Then, this coin must be reminted within time T since the creation or last remint of this coin.

In a general micropayment schemes, this process is performed at the bank. In our system, instead, the bank functionality is performed by a random but predefined set of users R_k. The selection of this set must be done in such a way that

- each user is responsible for reminting roughly the same amount of coins (load balance) and
- at least one honest user is a member of the remint set.

Whenever a user u has to remint a coin k, he sends it to each user in the remint set $R_k = \aleph_r(h(id(k)))$. Here, a collision resistant hash function h is used as a consistent mapping from the coin identifiers space to the user identifier space Π. Each user in R_k must verify the authenticity of k and store it in his local history database. If the verification succeeds, the reminters will create a multisignature

$$k_{new} = \{\texttt{XY}(k), ts, R_k, C_{R_k}, u\}_{R_k}$$

for the new coin with the same coin identifier and owner, but with a new time stamp ($\texttt{XY}()$ extracts the hash collision out of k). If the verification fails, either because the coin is invalid or because a coin with the same identifier and time stamp was already reminted, the reminters will audit the coin and trace back the cheater in the signature chain.

Protocol Description.

Minting	**Spending**
For a user u:	User u spends a coin at merchant m:
Initially: $K_u := \emptyset; sn_u := 0$	m : pick a nonce z
$sn_u := sn_u + 1$	$m \rightarrow u : z$
$ts := \texttt{now}()$	u : select $k \in K_u$
$x := u \vert\vert sn_u \vert\vert ts$	$u \rightarrow m : \{k, z, m, C_u\}_u \rightarrow k'$
Find y satisfying: $h_1(x) = h_2(y)$	$m : \mathsf{check}(m, k', z)$
$k := \langle x, y \rangle$	$u : K_u := K_u \backslash \{k\}$
$K_u := K_u \cup \{k\}$	$m : K_m := K_m \cup \{k'\}$

where $\texttt{now}()$ returns the current time and K_u is the bag of coins owned by user u

Reminting

User u remints a coin $k = \{\tilde{k}, z, u, C_s\}_s$:
$u : R = \aleph_r(h(id(k)))$
$u \rightarrow r_i : k, \texttt{now}(), R \rightarrow k', t', R' \quad \forall_i : r_i \in R$
$r_i : t' \approx \texttt{now}()$
$\quad R' = \aleph_r(h(id(k')))$
$\quad \mathsf{check}(u, k', \perp)$
$\quad \mathsf{verifyHistory}(k')$
$\quad k_{new} := \{\texttt{XY}(k'), t', R', C'_R, u\}$
$R \leftrightarrow u : \{k_{new}\}_R \rightarrow k_R$ (this is a three-round protocol)
$u : \mathsf{checkBase}(u, k_R)$

$\mathsf{check}(u, k, z)$:
if $\mathsf{isBase}(k)$ **then** $\mathsf{checkBase}(u, k)$
else $\{k', z', u', C_s\}_s := k$
$\quad$ **return** $(z' = z \vee z = \perp) \wedge u' = u$
$\quad\quad \wedge \mathsf{validSig}(k, s, C_s) \wedge \mathsf{check}(s, k', \perp)$

$\mathsf{checkBase}(u, k)$:
if $\mathsf{isReminted}(k)$ **then**
$\quad \{k', newts, R', C_R, u'\}_R := k$
$\quad$ **return** $u' = u \wedge R' = R = \aleph_r(h(id(k')))$
$\quad\quad \wedge newts \in [\texttt{now}() - T, \texttt{now}()]$
$\quad\quad \wedge \mathsf{validSig}(k, R, C_R)$
else
$\quad \langle x, y \rangle := k$
$\quad u'||sn||ts := x$
$\quad$ **return** $h_1(x) = h_2(y) \wedge u' = u$
$\quad\quad \wedge ts \in [\texttt{now}() - T, \texttt{now}()]$

$\mathsf{audit}(k, k')$:
$\{\ldots\{k_0, z_1, u_1, C_0\}_{u_0} \ldots, z_l, u_l, C_{l-1}\}u_{l-1} := k$
$\{\ldots\{k'_0, z'_1, u'_1, C'_0\}_{u'_0} \ldots, z'_{l'}, u'_{l'}, C'_{l'-1}\}u_{l'-1} := k'$
for $i = 1$ **to** $\mathsf{min}(l, l')$ **do**
    if $(z_i \neq z'_i \vee u_i \neq u'_i)$ **then return** u_{i-1}

$\mathsf{verifyHistory}(k)$:
$H_k := \{k' \in H |\ \mathtt{id}(k) = \mathtt{id}(k') \wedge (\neg\, \mathsf{isReminted}(k) \vee \mathtt{ts}(k) = \mathtt{ts}(k'))\}$
foreach $k' \in H_k$ **do**
    $B := B \cup \{\mathsf{audit}(k, k')\}$
$H := H \cup \{k\}$
return $H_k = \emptyset$

where B is the set containing all the blacklisted users and H is the set of all reminted coins, i.e., the remint history.

Security Analysis.

We will show that our protocol is secure by showing that for every double-spent coin, a corrupted node will get blacklisted.

Lemma 2.4.1. *Let r be the size of the remint set R. If $r > \gamma\eta + c$, for some constant γ, then the probability that R contains no honest nodes is less than $2^{-\eta}$.*

Proof. Since c nodes can be corrupted by the adversary at will, r must be larger than c which is ensured by the condition $r > \gamma\eta + c$. So we need to see how large the probability is that the remaining $r - c$ nodes happen to be taken form the remaining $t - c$ corrupted nodes when constructing the set R. We define a stochastic variable X equal to the number of honest nodes in R, given that c nodes in R are already corrupted. We want

$$\mathbb{P}(X = 0) \quad < \quad 2^{-\eta} \tag{2.0}$$

were η is our security parameter. As $c < r < n$ and $t < n$ we have

$$\begin{aligned}\mathbb{P}(X = 0) = \frac{\binom{t-c}{r-c}\binom{n-r+c}{c}}{\binom{n}{r}} &< \frac{\binom{t-c}{r-c}}{\binom{n-c}{r-c}} \\ &= \frac{(t-c)\dots(t-r+1)}{(n-c)\dots(n-r+1)} \\ &< \left(\frac{t-c}{n-c}\right)^{r-c}\end{aligned}$$

and we want

$$\begin{aligned}\left(\frac{t-c}{n-c}\right)^{r-c} = \left(\frac{n-c}{t-c}\right)^{c-r} &< 2^{-\eta} \\ (c-r)\log\left(\frac{n-c}{t-c}\right) &< -\eta \\ r &\geq \frac{\eta}{\log\left(\frac{n-c}{t-c}\right)} + c\,.\end{aligned}$$

Take $\gamma = \left(\log\left(\frac{n-c}{t-c}\right)\right)^{-1}$. Note that γ becomes independent of t and n when $t < \frac{n}{2}$. This completes the proof. □

Lemma 2.4.2. *Let k be a karma coin and $t = \mathtt{ts}(k)$. Then there is no relevant information (i.e., that leads to double-spending detection) in k after time $t + T$.*

Proof. The proof is split in two cases.

- If k is never double-spent in the period $[t, t + T]$ then there is no relevant information at all.
- If k is double-spent first at time t' with $t < t' < t + T$ then:
  - If k is reminted before t' then the new coin $\hat{k}$ with $\mathtt{ts}(\hat{k}) > t$ will contain relevant information to prove double-spending and therefore there is no relevant information in k.

– If k was not reminted before t' then both double-spent coins k_1 and k_2 must be reminted at least once before $t + T$(they would expire otherwise). Then any double-spending attested by k is detected before $t + T$.

□

Theorem 2.4.3. *Whenever a coin is double-spent, that coin expires or one corrupted node is identified.*

Proof. Whenever a coin is double-spent, both coins have the same identifier and time stamp. It is not possible for an adversary to change any of them: in case of a just minted coin they are protected by being part of the collision of the hash functions; and in the case of a re-minted coin it is not possible for an adversary to forge the multisignature, since Lemma 2.4.1 ensures that there is always a fair user in every remint set. Then, coins with the same identifier must be sent to the same remint set before their expiration time, otherwise they expire and the condition of the theorem holds. Therefore, at least one fair user $\dot{u}$ must receive both coins before their expiration time. Let k_{i_1} and k_{i_2} be the first remint request of each version of the double-spent coin k_i, received by $\dot{u}$ after the double-spending. Then, $\dot{u}$ detects fraud and calls $\texttt{audit}(k_{i_1}, k_{i_2})$. `audit` first checks whether the signatures are valid. It is clear that a user endorsing a coin with an invalid signature or that is improperly minted is faulty (he should have checked it). If that is not the case, then the coin is fairly minted and $id(k_{i_1}) = id(k_{i_2})$, at least the first user endorsing the coin is the same in k_{i_1} and k_{i_2}. Therefore, and given the fact that both coins are different, there must be one user in the signature chain that transferred the coin to two different users (or to the same user twice). In this case the user identities inside of the signature are different (or the challenge nonces are different), which constitutes a proof of double-spending. □

2.4.3 Handling Dynamic Networks

In a static network, the remint set R_k for a coin k never changes. That makes it easy to verify that a given coin was fairly reminted at some point in the past, as verifying a remint set is as trivial as checking $R_k = \aleph_r(h(id(k)))$.

In a dynamic network, it is not possible to be so restrictive while defining a valid remint set. Otherwise every time a user $u \in R_k$ is off-line, the coin k cannot be reminted, and therefore may expire. On the other hand, the selection of the users in R should somehow be predefined in order to limit the influence of the adversary, and at least allow the validity of the remint set to be reliably determined at a later time (unless we require $r > t$, which trivially implies that at least one fair node is in the remint set).

As a solution we define a valid remint set for a coin k, as the closest r users to $h(id(k))$ in the identifier space, that are on-line at remint time. Then the verification of the fairness of a remint set is difficult, given that the verifier has no information about the state of the network at remint time. An adversary could try to unfairly construct a remint set with only nodes that are under his control, by claiming that all other (fair) users were off-line at remint time. We are going to prevent this kind of attack by taking the dispersion of the remint set R as an indicator for the authenticity of the coin. We define the dispersion of a remint as

$$d(R_k) = \max_{i \in R_k} |i - h(id(k))| \ ,$$

assuming that closeness on the overlay network is equivalent to identifier difference.

Let us assume that the dispersion of the overlay network does not change very fast. Meaning that it is very unlikely, in a worldwide network with a large amount of users, to have big fluctuations in the amount of users connected to it, in short periods of time. Let α be the maximal rate of change for the dispersion of the overlay network over any interval of length at most σ, i.e. if T is the maximal time to live for a coin, and $d(t)$ is the dispersion at time t, then for all t' between t and $t + T$, we have $\frac{1}{\alpha} d(t) \leq d(t') \leq \alpha d(t)$.

We call a remint set *acceptable* (for a coin) if it satisfies our constraints on the remint set, and does not contain members beyond the boundaries specified by the dispersion. In such a scenario, an adversary does not have much freedom while selecting the users in R without drastically increasing $d(r)$.

Another issue that needs to be addressed in a dynamic network is the history transfer between users. The neighborhood R_k for a coin k should keep (as an invariant) the history of any reminted coin within the period $[\mathtt{ts}(k), \mathtt{ts}(k) + T]$. Given that the history consists of signed coins, it is not possible for an adversary to forge it. Therefore, a joining user can just renew its history by querying its neighbors for it and a leaving node has to broadcast his recent history to its neighborhood.

$\mathsf{checkBase}(u, k)$:
if $\mathsf{isReminted}(k)$ **then**
    $\{k', newts, R', C_R, u'\}_R := k$
    return $u' = u \wedge R' = R$
        $\wedge\, newts \in [\mathtt{now}() - T, \mathtt{now}()]$
        $\wedge\, d(R) \leq \alpha\, d(\mathtt{now}())$
        $\wedge\, \mathsf{validSig}(k, R, C_R)$
else
    $\langle x, y \rangle := k$
    $u'||sn||ts := x$
    return $h_1(x) = h_2(y)$
        $\wedge\, u' = u \wedge\, ts \in [\mathtt{now}() - T, \mathtt{now}()]$

The only modification that remains, with respect to the static version, is in the function checkBase, which now verifies that the dispersion of the remint set is not too big, instead of checking equality with the neighborhood.

Security Analysis.

We analyze security of the dynamic protocol similar to the static case.

Proposition 2.4.4 (Hoeffding bound)**.** *For a hyper-geometrically distributed random variable X, representing the number of successes among n draws, with probability p of success, we have [Hoe63, Chv79]*

$$\mathbb{P}(X \geq np + g) \leq e^{-2g^2/n}$$

Lemma 2.4.5. *Let $p = \frac{t-c}{n-c}$, fix β such that $c \leq \beta r$, and suppose $\beta + \alpha^2 p < 1$. If $r \in O\left(\frac{\alpha^2 \eta}{(1-\beta-p\alpha^2)^2}\right)$, then any acceptable remint set contains at least one honest node with probability $1 - 2^{-\eta}$.*

Proof. The remint set is fixed at remint time t. The adversary needs to pick r nodes for the remint set such that it does not violate the acceptability condition $d(R) \leq \alpha d(t')$, which is checked the next time the coin is reminted at time $t' \leq t + T$. At time t', the density $d(t') \leq \alpha d(t)$. This means that at time t the adversary can, at best, select r nodes among the $\alpha^2 r$ nodes with closer id to $h(id(k))$, for a coin k. Then he can still take control over c of them. Therefore, he succeeds if among these $\alpha^2 r$ nodes there are $r - c$ faulty ones.

Let X be a random variable representing the number of faulty nodes in such a sample of $\alpha^2 r$ nodes from all n nodes ($t - c$ of which are faulty). Then the adversary is successful if $X \geq r - c$. X is distributed according to the hyper-geometric distribution, with $p = \frac{t-c}{n-c}$, and we are interested in bounding

$$\begin{aligned}
\mathbb{P}(X \geq r - c) &\leq \mathbb{P}(X \geq r - \beta r) \qquad \{\beta + \alpha^2 p < 1\} \\
&= \mathbb{P}(X \geq p\alpha^2 r + (r - \beta r - p\alpha^2 r)) \qquad \{\text{Hoeffding bound}\} \\
&\leq e^{-2(r-\beta r - p\alpha^2 r)^2/\alpha^2 r} \\
&= e^{-2r(1-\beta-p\alpha^2)^2/\alpha^2}
\end{aligned}$$

which we want to be less than $2^{-\eta}$. Then, by taking logarithms

$$(-2r(1 - \beta - p\alpha^2)^2/\alpha^2) \log_2 e < -\eta$$

and hence

$$r \geq \frac{\alpha^2 \eta}{2(1-\beta-p\alpha^2)^2 \log_2 e}$$

which completes the proof. □

Lemma 2.4.6. *In every remint set, fair nodes can always transmit their remint history to another fair node before leaving.*

Proof. As a corollary of Lemma 2.4.5 and given the assumption that node joins and leaves are atomic operations, at least two fair nodes must be in a valid remint set, whenever a fair node is going to leave it. This fact, together with the secure routing assumption over the overlay network, implies that fair users can always transmit their remint history to another fair node before leaving. □

Theorem 2.4.7. *Let* $p = \frac{t-c}{n-c}$*, fix* β *such that* $c \leq \beta r$*, and suppose* $\beta + \alpha^2 p < 1$ *and* $r \in O\left(\frac{\alpha^2 \eta}{(1-\beta-p\alpha^2)^2}\right)$*. Then whenever a coin is double-spent, that coin expires or one corrupted node is identified with overwhelming probability.*

Proof. The proof of Theorem 2.4.3 also applies here provided that the remint set has at least one fair node and that the system is history preserving. The former condition follows from lemma 2.4.5 except with negligible probability on η. The latter condition follows from lemma 2.4.6. □

2.5 Early Double-spending Detection

In some scenarios double-spending detection might not be good enough. This is the case when an adversary is able to add new corrupted nodes easily. It is possible for a corrupted user who owns a karma coin, to spend it many times and very quickly, especially when the coin is just minted (or reminted). Although those actions are eventually going to be detected, this is not going to happen until the first two remint requests of this coin are submitted. This user of course is going to be punished, but then the

adversary might get another Id and repeat this operation. To counteract this kind of attacks, besides making it harder for an adversary to get new ids, it is possible to detect double-spending early. As a first line of defence, when a user receives a new coin, he performs a search over the coins he possess looking for duplicated identifiers. In case he succeeds, the double-spender is immediately blacklisted. The probability of finding a duplicated coin just like that is small, especially when the number of copies is not too big. To improve this, we introduce coin attractors to the system. An attractor is a user, whose hashed id is the closest to the hashed id of the coin. Then, when a user s wants to spend a coin at the merchant m, s searches over his coins for the one which has the minimum distance with the merchant's hashed id,

$$k_d = \min_{k \in K_s} |h(m) - h(id(k))| \ ,$$

and pays with it. Even thought faulty nodes may avoid sending coin to attractors, eventually a good node will do so. At that point the attractor will detect the double spending.

2.6 Conclusions

We have presented a completely decentralized, off-line karma implementation for P2P and grid systems, that detects double-spending and other types of fraud under varying adversarial scenarios. This is, so far, the first system for truly off-line karma coins, which can be used in highly dynamic peer-to-peer networks and grid systems. Our system has smaller message complexity than previously proposed systems of similar characteristics, under certain scenarios. In particular, we are able to completely replace a central bank by a distributed remint set whose size is roughly proportional to the security parameter η.

Chapter 3

Protocol Analysis of Anonymity in the Formal Setting

3.1 Introduction

There is a growing concern about the amount of personal information that is being gathered about people's actions. Websites habitually track people's browsing behavior, banks track people's financial whereabouts, and current trends in applications of RFID chips will make it possible to track people's physical location. Furthermore, several European governments oblige Internet Service Providers to keep traffic logs of all communication on the Internet.

To protect users from undesired information leaks, there are various protocols and tools that provide some form of anonymity, e.g., for web browsing and e-mailing; for instance Tor [DMS04], Freenet [CSWH01], and Mixminion [DDM03].

Anonymizing Protocols In 1981, Chaum [Cha81] pioneered the idea of what are currently called *Chaum mixes*. If A wants to send a message m to B, she chooses a number of relays, say R_1, R_2, and sends a message

$\{\!|R_2, \{\!|B, \{\!|m|\!\}_B|\!\}_{R_2}|\!\}_{R_1}$ to R_1. Here $\{\!|\ldots|\!\}_X$ denotes encryption with the public key of agent X. Relay R_1 decrypts the first layer of encryption and forwards the message $\{\!|B, \{\!|m|\!\}_B|\!\}_{R_2}$ to R_2, who peels off another layer and sends the remainder to B. Traffic analysis is complicated as every relay queues messages until a certain number have been received and relays them in a batch. Also, dummy network traffic can be generated if not enough messages are available. This technique forms the basis of Onion Routing [GRS96] and its successor Tor [DMS04], a currently running network of relays providing anonymous communication for instance with web servers.

Another idea to provide some kind of anonymity was proposed by Reiter and Rubin in [RR98]; the idea is to use so called crowds. The message m is sent to a relay R_1, which then probabilistically either forwards the message to another relay R_2, or to its final destination. A key idea here is that every member of the network can act as a relay.

Various notions of anonymous communication exist and various attacker models have to be considered. Chaum mixes typically provide *sender anonymity*: the ultimate receiver B of the message m does not know who originated the message. It also provides *unlinkability*: someone observing all network traffic cannot tell that A and B are communicating.

Crowds does not provide unlinkability: a global eavesdropper will be able to see the message m passing through the whole network and will see the message m moving from A to B over a number of relays. However, it does provide sender anonymity even if some of the members of the network are corrupted.

Formal methods For security and authentication, a number of formal methods and automated tools have been developed to determine the correctness of security and authentication protocols (see, e.g., [AG97, Pau98, JHar, Gut01]). In contrast, the formalization of anonymity is still in its infancy. This chapter presents a formal framework in which various information hiding properties can be expressed in a clear and precise manner. It enables us to formulate competing notions of anonymity in a uniform fashion and compare them.

Our contribution and related work Starting point of our approach,

also present in [HO05], is the idea that various information hiding properties can best be expressed in the language of *epistemic logic* [FHMV95]. As all these properties are related to what principals know, this approach arises naturally. Epistemic (modal) logic reasons about what agents know and believe. In order to do so, the reasoning capacities of the agents are idealized. For instance, it is assumed that agents never forget anything and that they extract as much information as possible from what they see. In this way, clear inference rules can be formulated. Then, it is possible to reason about what each principal knows at a specific point in time.

This makes it possible to reason not only about the messages in a run, but also about the knowledge agents gain from these messages. For instance, sender anonymity can typically be formulated as the receiver not knowing who sent a certain message.

Central in the epistemic framework is our notion of observational equivalence of runs. An agent knows a fact of a certain run r if that fact is true in all *possible worlds*, i.e., in all runs that are observationally equivalent to r. In [HO05], the observational equivalence is basically assumed, whereas we actually construct such a relation.

Our notion of observational equivalence takes care of the cryptographic operations that are defined on the set of messages.

We should remark that other works present similar equivalence relations on the set of messages, e.g., [MVdV04, AR02, MVdV07]. Ours is unique in the sense that it not only deals with whether or not messages are encrypted, but also takes care of relations between ciphertexts and hash values. Especially for dealing with unlinkability this is essential. For example, if an agent A sends a message $\{\!|m|\!\}_k$ and another agent B either sends the message $\mathsf{hash}(\{\!|m|\!\}_k)$ (for instance, to acknowledge receipt of A's message), or the message $\{\!|m|\!\}_k$ (for instance, forwarding it to someone else), then a global observer could *link* A and B, even though all messages involved look like random bitstrings to this global observer.

Note that this formal approach is possibilistic in nature: it only considers whether or not it is possible that A and B are communicating, but not the probability that they are. There is another body of literature studying probabilistic aspects of anonymity, e.g., [Shm04, Ser04]. These papers gen-

erally *assume* the correctness of the anonymizing protocols in the formal message sense and then compute measures of anonymity.

We will show that this epistemic framework allows us to formally describe and verify various information hiding properties that are considered in the literature.

Organization of this chapter We start in Section 3.2 with the definition of the message algebra. Then Section 3.3 fixes the network and intruder model and describes the fundamental notion of observational equivalence of runs. Section 3.4 introduces the epistemic operators and shows how these operators can be used to express various information hiding properties, such as sender anonymity and unlinkability. Finally, to illustrate the power of our approach, we consider in Section 3.5 abstract versions of Crowds and Onion Routing and formally prove their anonymizing properties. We also consider a version of Onion Routing that has a subtle error and show how our framework is capable of detecting this flaw.

Notations and terminology Throughout this chapter a secret key is called a *symmetric key*. In an asymmetric encryption a key pair consists of a *public key* and a *private key*.

We clearly distinguish the verbs *to know* and *to possess*: an agent *knows a fact* while it *possesses a message*. The former acts as an epistemic operator which is applied to a proposition on the run, while the latter concerns accessibility to messages via cryptographic operations such as encryption or decryption.

The set X^* consists of all the finite lists over X, that is, $X^* = \coprod_{n\in\mathbb{N}} X^n$. For a list r with length $|r|$, the i-th element of r is denoted by r_i. The head-closed sublist $\langle r_0, r_1, \ldots, r_{i-1}\rangle$ of r is denoted by $r_{<i}$.

3.2 The message algebra

In this section we present our model of the message algebra, and introduce the closure operator on a set of messages. This is done in a standard way similar to [Pau98, JHar].

Definition 3.2.1. (Message algebra) **Agent** is an infinite set of *agents identifiers*, **Key** is an infinite set of *key symbols*, **Nonce** an infinite set of *nonce symbols*. Agents are denoted by $A, B, \ldots$, keys are denoted by $k, k', \ldots$, nonces by $n, n', \ldots$. Using these building blocks, *messages* are constructed using encryption, hashing, and pairing operations:

$$\mathbf{Msg} \ni m := A \mid n \mid k \mid \mathsf{enc}(m, k) \mid \mathsf{hash}(m) \mid \langle m, m \rangle.$$

We also assume that there is a function $(-)^{-1}$ from keys to keys with the property that $(k^{-1})^{-1} = k$. If $k^{-1} = k$, then k is called a *symmetric* key, otherwise one of k and k^{-1} is called a *private* key and the other one a *public* key.

We do not include a specific type for data, as it can be modeled using nonces. This is straightforward as we need to define the 'identity of a message' and using nonces we get that for free.

Remark 3.2.1. Throughout this chapter we abstract from subtle issues regarding the length of the ciphertexts. In practical implementations of these protocols it is necessary to take care of this issue. The general guideline is that all encryptions should look alike to the adversary. This can be achieved using padding. For instance, as it is done in some variants of onion routing: take the maximum-length ciphertext (which should be constant, as a system requirement) of the protocol and make sure that every other ciphertext has that length, by concatenating a random bitstring at the end.

We abbreviate $\mathsf{enc}(m, k)$ to $\{\!|m|\!\}_k$ and omit angle brackets inside encryptions. Typically, an agent A has an associated public key and we denote encryption with the public key of A simply by $\{\!|m|\!\}_A$. Terms in $\mathbf{Agent} \cup \mathbf{Nonce} \cup \mathbf{Key}$ are said to be *primitive*.

Definition 3.2.2 (Sub-message)**.** Define the relation sub-message, notation $\preceq$, as the smallest reflexive and transitive relation satisfying

1. $m_1 \preceq \langle m_1, m_2 \rangle$
2. $m_2 \preceq \langle m_1, m_2 \rangle$

3. $m \preceq \{\!|m|\!\}_k$
4. $m \preceq \mathsf{hash}(m)$.

Note that this relation is unaware of cryptographic operations; e.g., $m \preceq \{\!|m|\!\}_k$ and $m \preceq \mathsf{hash}(m)$ always hold.

Furthermore, a function $\pi\colon \mathbf{Msg} \to \mathbf{Msg}$ is said to be *structure preserving* if it maps names to names, nonces to nonces, keys to keys, encrypted messages to encrypted messages, hashes to hashes, and satisfies $\pi(\langle m_1, m_2\rangle) = \langle \pi(m_1), \pi(m_2)\rangle$ for all messages m_1 and m_2. Note that $\pi(\mathsf{hash}(m)) = \mathsf{hash}(m')$ but not necessarily with $m' = \pi(m)$. Similarly for encryption.

The closure $\overline{U}$ of a set U of messages consists of all the messages that can be extracted and constructed from those in U using cryptographic operations such as decryption, encryption, tupling and decomposing tuples. The formal definition follows

Definition 3.2.2 (Closure)**.** Let U be a set of messages. The *closure* of U, denoted by $\overline{U}$, is the smallest set of messages satisfying:

1. $U \subseteq \overline{U}$;
2. if $\{\!|m|\!\}_k, k^{-1} \in \overline{U}$, then $m \in \overline{U}$;
3. if $\langle m_1, m_2\rangle \in \overline{U}$, then $m_1, m_2 \in \overline{U}$;
4. if $m, k \in \overline{U}$, then $\{\!|m|\!\}_k \in \overline{U}$;
5. if $m \in \overline{U}$, then $\mathsf{hash}(m) \in \overline{U}$;
6. if $m_1, m_2 \in \overline{U}$, then $\langle m_1, m_2\rangle \in \overline{U}$.

This closure operation is important here for two reasons. One reason, which is common in formal modelling of security protocols, is that an agent may send only the messages he can construct from what he has seen (see Definition 3.3.4). The other one is that the closure of the possessions of an agent restricts the set of runs that the agent considers to be observationally equivalent to a given run (see Definition 3.3.8).

3.3 Runs and observational equivalence

3.3.1 Network model

To model anonymity properties, we have to be able to talk about the *sender* and *receiver* of a message. Therefore we include explicit sender and receiver information in our notion of *event* below. In a real world setting, an event would correspond to an IP package, which has sender and receiver information in the header and the actual message in the body.

Definition 3.3.1. (Agents, events) We denote by **AG** a non-empty set of agents, which has a special element spy for the intruder. An agent which is not spy is called a *regular* agent. An *event* is a triple (A, B, m) of the *sender* A, the *recipient* B and the *delivered message* m. To make the intention clear we denote the above event by $(A \to B : m)$. The set of all events is denoted by **Event**.

Definition 3.3.2. (Runs) A *run* is a finite list of events, i.e. an element of the set $\mathbf{Run} := \mathbf{Event}^*$. A run describes all the events that have occurred in a network. The function $\mathsf{msg}\colon \mathbf{Run} \to \mathcal{P}(\mathbf{Msg})$ extracts all the messages occurring in a run. That is,

$$\begin{aligned}\mathsf{msg}\big(\ & (A_0 \to B_0 : m_0),\\ & (A_1 \to B_1 : m_1),\\ & \quad\vdots\\ & (A_{n-1} \to B_{n-1} : m_{n-1})\ \big) = \{m_0, m_1, \ldots, m_{n-1}\}.\end{aligned}$$

3.3.2 Attacker model

It is standard to verify security properties such as authentication or secrecy under the Dolev-Yao intruder model [DY83]. In that model the network is completely under the control of the intruder, hence no agent can be sure who sent the message it receives, or whether the intended recipient actually receives the message sent.

To model anonymity properties, it is customary to use a weaker attacker model. We assume that the intruder is *passive*, i.e., *observes* all network traffic, but does not actively *modify* it. It is however possible that some regular agents are *corrupted*; we will later model this in terms of the keys the intruder possesses. This setting enables us to express that a run of a protocol satisfies a certain information hiding property as long as certain agents are not corrupted. Contrary to the intruder, the regular agents are not necessarily aware of all the events in the run; we adopt the convention that they only see the events in which they are involved as sender or receiver.

Definition 3.3.3. (Visible part of runs) Let r be a run. For a regular agent $A \in \mathbf{AG} \setminus \{\mathsf{spy}\}$ the *A-visible part* of r, denoted by $r|_A$, is the sublist of r consisting of all the events that have A in either sender or receiver field. The spy-*visible part* of r, denoted by $r|_{\mathsf{spy}}$, is identical to r.

3.3.3 Communication protocol

It is important to specify the set of runs *of a protocol*, which is the set of possible worlds used in the definition of knowledge. There, runs which are illegitimate with respect to a protocol are excluded because every agent knows that those runs cannot happen. In that sense, a protocol is *common knowledge*. In this chapter a protocol consists of two components, namely the *candidate runs* and the *initial possessions*.

The set of candidate runs is just a set of runs. Intuitively, it consists of those runs which fulfill the requirements on which messages may/must be sent by an agent. In the examples (see Section 3.5) we describe the set of candidate runs.

An initial possession function $\mathsf{IPo} : \mathbf{AG} \to \mathcal{P}(\mathbf{Msg})$ assigns to each agent the set of messages the agent possesses before communication takes place. Typically an agent's initial possessions consists of its private key and the public keys of all agents. We require that, for every agent A, $\mathsf{IPo}(A)$ consists of only primitive messages.

We denote by $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$ the set of the messages that an agent A *possesses* at stage i of the run r. It is the closure of the union of: 1)

A's initial possession $\mathsf{IPo}(A)$; 2) the messages A has seen up to this point, $\mathsf{msg}\big((r_{<i})\ |_A\big)$; 3) the nonces and secret keys A has freshly generated at stage i. The set $\mathsf{Poss}_{\mathsf{IPo}}(r, A, i)$ consists of all messages the agent A can possibly construct after having seen the first i messages of the run.

Definition 3.3.4. A run $r \in \mathbf{Run}$ is said to be *legitimate* with respect to an initial possession function $\mathsf{IPo}\colon \mathbf{AG} \to \mathcal{P}(\mathbf{Msg})$ if, for every $i \in [0, |r| - 1]$, $m_i \in \mathsf{Poss}_{\mathsf{IPo}}(r, A_i, i)$, where $(A_i \to B_i : m_i) = r_i$.

Definition 3.3.5. (Protocols and runs of protocol) A *protocol* is a pair $(\mathsf{cr}, \mathsf{IPo})$ consisting of a set of candidate runs cr and an initial possession function IPo. A run $r \in \mathbf{Run}$ is said to be a *run of a protocol* $(\mathsf{cr}, \mathsf{IPo})$ if $r \in \mathsf{cr}$ and is legitimate with respect to the initial possession function IPo. This captures the notion that principals cannot guess keys or nonces they have not seen or generated. The set of runs of a protocol $(\mathsf{cr}, \mathsf{IPo})$ is denoted by $\mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$.

Note that in the literature (e.g., [FHMV95]) a protocol is often given locally, by specifying what kind of messages an agent can send in a certain situation. For our theory, the way of specifying a protocol is unimportant; therefore we abstract from this specification by considering a given set of candidate runs.

3.3.4 Observational equivalence

If an agent receives an encrypted message for which it does not have the decryption key, this message looks just like a random bitstring. This section formalizes this idea by defining the notion of observational equivalence. Intuitively, two sequences of messages look the same to an agent if they are the same for the messages the agent understands and if a message in one sequence looks like a random bitstring to the agent, then the corresponding message in the other sequence also looks like a random bitstring. Matters are slightly more complicated: we have to take care of the case where a specific random looking bitstring occurs more than once in a sequence of messages; we have to take care of the possibility for someone to understand

only a submessage of a message; we also have to take care of the case where certain messages look like random bitstrings to the agent, but are still somehow related. For example, if the agent does not possess the symmetric key k, two messages $m_1 = \{\!|m|\!\}_k$ and $m_2 = \mathsf{hash}(\{\!|m|\!\}_k)$ both look like random bitstrings. Still the agent can derive a relationship between them, namely $m_2 = \mathsf{hash}(m_1)$.

This motivates our definition of observational equivalence below. The definition is typically applied with U being equal to $\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r| - 1)$, the set of messages that an agent A possesses after the run has finished.

Definition 3.3.6. (Reinterpretations of messages) Let π be a structure preserving permutation on the set **Msg** of messages and let $U \subseteq$ **Msg** be a set of messages. The map π is said to be a *semi-reinterpretation under* U if it satisfies the following conditions:

$$\begin{aligned} \pi(p) &= p && \text{for primitive terms } p \\ \pi(\{\!|m|\!\}_k) &= \{\!|\pi(m)|\!\}_k && \text{if } m,\ k \text{ are in } U, \text{ or} \\ & && \text{if } \{\!|m|\!\}_k,\ k^{-1} \text{ are in } U \\ \pi(\mathsf{hash}(m)) &= \mathsf{hash}(\pi(m)) && \text{if } m \text{ is in } U. \end{aligned}$$

The map π is called a *reinterpretation under* U if it is a semi-reinterpretation under U and if π^{-1} is a semi-reinterpretation under $\pi(U)$ as well.

The above definition says that as far as an agent can observe (i.e., for all the messages in U), the permutation π preserves structural and cryptographic relationships between messages. We extend reinterpretations naturally to events (applying π to the message field of an event) and to runs (applying π to the message field of every event in a run).

Lemma 3.3.7. *Let U be a set of messages.*

1. *The identity map on the set of messages is a reinterpretation under U.*
2. *For every reinterpretation π under U, its inverse π^{-1} is a reinterpretation under $\pi(U)$.*

3. *For all reinterpretations π under U and π' under $\pi(U)$, their composition $\pi' \circ \pi$ is a reinterpretation under U.*

Proof. Trivial. Note that 3 is already true for semi-reinterpretations. □

Definition 3.3.8. (Observational equivalence of runs) Let $r, r' \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$ be two runs of a protocol $(\mathsf{cr}, \mathsf{IPo})$ and let $A \in \mathbf{AG}$ be an agent. Two runs r and r' are said to be *observationally equivalent for an agent* A, denoted by $r \cong_A r'$, if there exists a reinterpretation π under $\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r|-1)$ such that $\pi(r|_A) = r'|_A$. Such a reinterpretation will be called a *reinterpretation for* A.

Lemma 3.3.9. *For each agent $A \in \mathbf{AG}$, the relation $\cong_A$ on $\mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$ is an equivalence relation.*

Proof. This follows from Lemma 3.3.7. Note that $\pi\big(\mathsf{Poss}_{\mathsf{IPo}}(r, A, |r|-1)\big) = \mathsf{Poss}_{\mathsf{IPo}}(\pi(r), A, |r|-1)\big)$ since $\mathsf{IPo}(A)$ consists solely of primitive messages and since a reinterpretation does not change the sender and receiver fields of events. □

Example 3.3.10. Consider the following two runs in which A and A' are sending the messages m and m' to B via a single relay, or Chaum mix, M. Below n and n' are fresh nonces.

r	r'
$(A \to M : \{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M)$	$(A \to M : \{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M)$
$(A' \to M : \{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M)$	$(A' \to M : \{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M)$
$(M \to B : \{\!\|m\|\!\}_B)$	$(M \to B : \{\!\|m\|\!\}_B)$
$(M \to B : \{\!\|m'\|\!\}_B)$	$(M \to B : \{\!\|m'\|\!\}_B)$

Assume that the spy knows the identities of all agents, all public keys and also the private key of B. Then these runs are still observationally equivalent for the spy; we can take a reinterpretation that exchanges $\{\!\|n, B, \{\!\|m\|\!\}_B\|\!\}_M$ and $\{\!\|n', B, \{\!\|m'\|\!\}_B\|\!\}_M$ and leaves $\{\!\|m\|\!\}_B$ and $\{\!\|m'\|\!\}_B$ fixed. It is important to realize that a reinterpretation π for the spy must satisfy $\pi(\{\!\|m\|\!\}_B) = \{\!\|\pi(m)\|\!\}_B$ (since B's private key is compromised), but does not

necessarily have to satisfy $\pi(\{\!|n, B, \{\!|m|\!\}_B|\!\}_M) = \{\!|n, B, \pi(\{\!|m|\!\}_B)|\!\}_M$ (and similarly for m').

Without the nonces, however, the runs are not observationally equivalent.

r	r'
$(A \to M : \{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M)$	$(A \to M : \{\!\|B, \{\!\|m'\|\!\}_B\|\!\}_M)$
$(A' \to M : \{\!\|B, \{\!\|m'\|\!\}_B\|\!\}_M)$	$(A' \to M : \{\!\|B, \{\!\|m\|\!\}_B\|\!\}_M)$
$(M \to B : \{\!\|m\|\!\}_B)$	$(M \to B : \{\!\|m\|\!\}_B)$
$(M \to B : \{\!\|m'\|\!\}_B)$	$(M \to B : \{\!\|m'\|\!\}_B)$

This is because π must satisfy $\pi(\{\!|B, \{\!|m|\!\}_B|\!\}_M) = \{\!|B, \pi(\{\!|m|\!\}_B)|\!\}_M$ (and similarly for m'). Note how this nicely captures the fact that, from the viewpoint of the spy, the messages $\{\!|B, \{\!|m|\!\}_B|\!\}_M$ and $\{\!|B, \{\!|m'|\!\}_B|\!\}_M$ can indeed be distinguished: at the end of the run the spy possesses the messages $\{\!|m|\!\}_B$ and $\{\!|m'|\!\}_B$ and he can simply encrypt the messages $\langle B, \{\!|m|\!\}_B\rangle$ and $\langle B, \{\!|m'|\!\}_B\rangle$ with the public key of M. This does not hold for the version with the nonces, since the spy never possesses these nonces.

3.4 Formulas and epistemic operators

In this section we introduce the epistemic (or modal) language as our specification language. This language will alow us to express security notions that we can later analyze. The semantics of epistemic operators is defined in a standard way [FHMV95], taking the set $\mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of runs of a protocol as the set of possible worlds equipped with observational equivalence $\cong_A$. Note that our language is *semantics-based*: a formula is identified as a $\{\mathbf{T}, \mathbf{F}\}$-valued function over a model, rather than a syntactically-defined entity.

3.4.1 Formulas

With a formula we want to express not only a fact about a run but also that an agent knows/does not know a certain fact about a run.

Definition 3.4.1. (Formulas) A *formula* φ is a function which takes as its arguments a protocol $(\mathsf{cr}, \mathsf{IPo})$ and a run $r \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of that protocol, and returns either $\mathbf{T}$ or $\mathbf{F}$. The set of all the formulas is denoted by $\mathbf{Form}$. We follow the tradition of logic to denote the fact $\varphi(\mathsf{cr}, \mathsf{IPo}, r) = \mathbf{T}$, where $r \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$, by $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$. Often the protocol $(\mathsf{cr}, \mathsf{IPo})$ under consideration is clear from the context, in which case we abbreviate $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$ to $r \models \varphi$. Logical connectives on formulas such as $\wedge$, $\vee$, $\rightarrow$ and $\neg$ are defined in an obvious way. A formula φ is said to be *valid* if $\mathsf{cr}, \mathsf{IPo}, r \models \varphi$ for all $r \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$.

We will use the following abbreviations Sends, Possesses, and Originates repeatedly in the rest of this chapter. They express fundamental properties of runs.

Definition 3.4.2. The formula A Sends m to B means: at some stage in the run, A sends a message to B which *contains* m *as a subterm.*

$$r \models A \text{ Sends } m \text{ to } B \quad \stackrel{\text{def}}{\Longleftrightarrow}$$
$$\exists i \in [0, |r| - 1].\ \big(m \preceq m' \text{ where } (A \rightarrow B : m') = r_i\big).$$

We will also use A Sends m to mean that A sends the message m to someone.

$$r \models A \text{ Sends } m \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad \exists B.\ r \models A \text{ Sends } m \text{ to } B.$$

The formula A Possesses m means: *after the run has finished*, A is capable of constructing the message m.

$$r \models A \text{ Possesses } m \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad m \in \mathsf{Poss}_{\mathsf{IPo}}(r, A, |r| - 1).$$

The formula A Originates m means that: A Sends m, and m contains a fresh nonce or key.

$$r \models A \text{ Originates } m \quad \stackrel{\text{def}}{\Longleftrightarrow}$$
$$\exists i \in [0, |r| - 1]. \exists l \in \mathbf{Nonce} \cup \mathbf{Key} \Big(l \preceq m \preceq m' \text{ where } (A \rightarrow B : m') = r_i$$
$$\wedge \forall j \in [0, i - 1].\ \big(l \not\preceq \hat{m} \text{ where } (A' \rightarrow A : \hat{m}) = r_j\big)\Big).$$

3.4.2 Epistemic operators

Using the observational equivalence relations over the set $\mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$ of possible worlds defined in Section 3.3.4, we can now introduce epistemic operators in the standard way (see e.g., [FHMV95]).

Definition 3.4.3. (Epistemic operators) Let $(\mathsf{cr}, \mathsf{IPo})$ be a protocol. For an agent $A \in \mathbf{AG}$, the epistemic operator $\Box_A : \mathbf{Form} \to \mathbf{Form}$ is defined by:

$$\mathsf{cr}, \mathsf{IPo}, r \models \Box_A \varphi \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad \forall r' \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}.\ \big(r' \cong_A r \implies \mathsf{cr}, \mathsf{IPo}, r' \models \varphi\big).$$

The formula $\Box_A \varphi$ is read as "after the run is completed, the agent A *knows* that φ is true". The formula $\Diamond_A \varphi$ is short for $\neg\Box_A\neg\varphi$ and read as "after the run is completed, the agent A *suspects* that φ is true".

Lemma 3.4.4 ($\Box_A$ is the **S5**-modality)**.** *For each agent $A \in \mathbf{AG}$, the operator $\Box_A$ satisfies the following properties.*

- *(Necessitation) For each set of candidate runs* cr *and each initial possession function* IPo*, if* $\big(\forall r \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}.\ \ r \models \varphi\big)$*, then* $\big(\forall r \in \mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}.\ \ r \models \Box_A\varphi\big)$*.*
- *The following formulas are all valid: (Distribution)* $\Box_A(\varphi \to \psi) \to (\Box_A\varphi \to \Box_A\psi)$*; (T)* $\Box_A\varphi \to \varphi$*; (4)* $\Box_A\varphi \to \Box_A\Box_A\varphi$*; (5)* $\Diamond_A\varphi \to \Box_A\Diamond_A\varphi$*.*

In short, the operator $\Box_A$ is a so-called **S5***-modality.*

Proof. Since the epistemic operator $\Box_A$ is defined via an equivalence relation $\cong_A$ on $\mathbf{Run}_{\mathsf{cr},\mathsf{IPo}}$. See e.g., [BdRV01]. □

3.4.3 Expressing information hiding properties

There are a number of properties which are referred to as information hiding properties (see e.g., [PK00]), and the meaning of some of these properties

might be different from one application to another. As stated in the introduction, we aim at, rather than a decisive definition of "anonymity", a framework in which we can formulate and analyze various different properties in a uniform and straightforward manner.

In this subsection we formulate some common examples of information hiding properties in our epistemic language. We use the standard notion of an *anonymity set*: it is a collection of agents among which a given agent is not identifiable. The larger this set is, the more anonymous an agent is.

Sender anonymity Suppose that r is a run of a protocol in which an agent B receives a message m. We say that r provides *sender anonymity* with anonymity set **AS** if it satisfies

$$r \models \bigwedge_{X \in \mathbf{AS}} \Diamond_B(X \text{ Originates } m).$$

This means that, as far as B is concerned, every agent in the anonymity set could have sent the message.

Unlinkability We say that a run r provides *unlinkability* for users A and B with anonymity set **AS** if

$$r \models \big(\neg\Box_{\mathsf{spy}}\varphi_0(A,B)\big) \wedge \bigwedge_{X \in \mathbf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)\ ,$$

where $\varphi_0(X,Y) = \exists n.\ \big(X \text{ Sends } n \wedge Y \text{ Possesses } n\big)$. Intuitively, the left side of the conjunction means that the adversary is not certain that A sent something to B. The right side means that every other user could have sent something to B. Similarly, unlinkability between a user A and a message m could be defined as $\models \neg\Box_{\mathsf{spy}}(A \text{ Sends } m) \wedge \bigwedge_{X \in \mathbf{AS}} \Diamond_{\mathsf{spy}}(X \text{ Sends } m)$.

Plausible deniability In certain circumstances (e.g., relays), agents might be interested in showing that they did not know that they had some sensitive information m. This might be modelled by the following epistemic formula:

$$r \models \Box_{\mathsf{spy}}\neg(\Box_A(A \text{ Possesses } m))\ .$$

This formula is read as: the spy knows that A does not know that she possesses m. i.e., there is a run in which A does not possess m.

3.5 Examples

To illustrate the applicability of the theory, we analyze the anonymity of simplified versions of Crowds and Onion Routing. For Crowds, we focus on proving sender anonymity, whereas for Onion Routing we are concerned with unlinkability.

3.5.1 Crowds

The Crowds system [RR98] is a system for doing anonymous web transactions based on the idea that anonymity can be provided by hiding in a crowd. Next we describe a simplified version of it. When someone wants to send a request to a server, she randomly selects a user from a crowd of users and asks this user to forward the request for her to the server. This user then either forwards the request to the server, or selects another random user from the crowd to do the forwarding. (Note: this only describes the request part; not the reply part). For simplicity reasons, we model requests as nonces.

A Crowds node A might perform any of the following actions

- at any time A can send a request n for server S to a relay B: $(A \rightarrow B : \langle S, n \rangle)$, where n is freshly generated;
- when a message of the form $(B \rightarrow A : \langle S, n \rangle)$ is received, A can either deliver it to destination: $(A \rightarrow S : n)$ or forward it to a randomly-chosen relay C: $(A \rightarrow C : \langle S, n \rangle)$.

This obviously provides sender anonymity, in the sense that the server cannot tell from who the request really originated. In the formal framework this can be formulated as follows:

Theorem 3.5.1. *Let r be a run of Crowds in which $(A \rightarrow S : n)$ occurs with $A \neq S$. Then*

$$r \models \bigwedge_{B \in \mathbf{AS}} \Diamond_S(B \text{ Originates } n),$$

where the anonymity set **AS** *is equal to* $\mathbf{AG} \setminus \{S\}$.

Proof. Let $B \in \mathbf{AS}$ and take $r' = (B \to A : n), r$, i.e., r augmented by a fictitious event $(B \to A : n)$. Then r' is also a valid run and obviously $r' \models B$ Originates n. Because $S \neq A, B$, we have $r'|_S = r|_S$ and therefore $r' \cong_S r$. So $r \models \Diamond_S(B \text{ Originates } n)$. □

Against a global eavesdropper, i.e., someone who can observe *all* network traffic, this does not provide anonymity. This is, of course, also remarked in [RR98].

Theorem 3.5.2. *Let r be a run of Crowds in which A freshly sends n to S over a relay R. I.e., the run contains*

$$(A \to R : \langle S, n \rangle)$$

and the nonce n is fresh in $A \to R : \langle S, n \rangle$. Then $r \models \Box_{\mathsf{spy}}(A \text{ Originates } n)$.

Proof. Suppose that $r \models \neg\Box_{\mathsf{spy}}(A \text{ Originates } n)$. Then there is a run r' satisfying $r \cong_{\mathsf{spy}} r'$ where the formula $r' \models \neg(A \text{ Originates } n)$ holds. Let π be the reinterpretation for spy such that $\pi(r) = r'$. Note that $\pi(n) = n$ and $\pi(S) = S$. Since A did not originate n in r', there must be an event $B \to A : m$ with n as subterm appearing in m before the event $A \to R : \langle S, n \rangle$. This message m must be of the form n or of the form $\langle C, n \rangle$ for some agent C, since the protocol does not allow the sending of messages of any other form and since it must have n as subterm. However, this means that $B \to A : \pi^{-1}(m)$ occurs in r before $A \to R : \langle S, n \rangle$. Note that $\pi^{-1}(m)$ is of the form $\pi^{-1}(n) = n$ or of the form $\pi^{-1}(\langle C, n \rangle) = \langle C, n \rangle$ for some agent C. This contradicts the fact that n appears freshly in the event $A \to R : \langle S, n \rangle$. □

3.5.2 Onion Routing

Chaum mixes were already mentioned in the introduction. Users construct messages of the form $\{\!|R_2, \{\!|B, \{\!|m|\!\}_B|\!\}_{R_2}|\!\}_{R_1}$ (called *onions*, because of the layered encryptions) and use relays (R_1, R_2) that decrypt one layer (*peel the onion*) and forward the remainder. In this section we analyze the unlinkability of a very simple version.

An agent in the set **AG** belongs to one of the following families:

1. *Onion routers* who try to disguise causal relations between incoming and outgoing messages by peeling onions (i.e., removing one layer of encryption) and forwarding them. The router collects incoming messages until it has received k messages. Then, the messages are permuted and sent in batch to their respective intended destinations.
2. *Users* who try to communicate with one another unlinkably, with the help of the onion router.
3. The *intruder* denoted by spy.

In the sequel we assume that there is only a single router denoted by M (note that M is of type router, in contrast with users $X, Y, \dots$). Furthermore the initial possession function IPo is such that, for each agent X, $\mathsf{IPo}(X)$ consists of the private key of X and the public keys of all agents. For now, no agent is corrupt, so also spy does not possess any private keys other than his own.

For a user X there are two actions that can be performed at any time

- X can initiate sending a fresh message n to another user Y by building an onion and sending it to the router M: $(X \to M : \{\!|n_0, Y, \{\!|n|\!\}_Y|\!\}_M)$, with n_0 fresh;
- or X may also send random messages to the router M, padding the network with dummy messages. These messages will be ignored by the router; they only serve to obscure the relation between incoming and outgoing messages if not enough real traffic is available: $(X \to M : \{\!|n|\!\}_M)$.

The router M will wait until he has received k messages, namely: l onions of the form $(X_i \to M : \{\!|n_i, Y_i, m_i|\!\}_M)$ and $k-l$ padding messages of the form $(X_j \to M : \{\!|n_j|\!\}_M)$ with $i = 1 \dots l, j = l+1 \dots k$ and X_i, X_j, Y_i pairwise distinct. Then it "peels" the onions and forwards the messages to the intended recipients in random order: $(M \to Y_i : m_i)$ with $i = 1 \dots l$. Note that l is a free variable and can be instantiated by any value $1 \leq l \leq k$, whereas k is a constant.

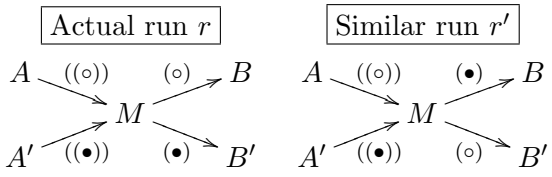

Figure 3.1: Onion Routing

Next, we investigate under what conditions onion routing ensures unlinkability, that is, which runs r of onion routing make the formula $\neg\Box_{\mathsf{spy}}\varphi_0(A,B) \wedge \bigwedge_{X\in\mathbf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$ true. Remember that we abbreviate the formula $\exists n.\ \big(X \mathsf{\ Sends\ } n \wedge Y \mathsf{\ Possesses\ } n\big)$ to $\varphi_0(X,Y)$.

Example 3.5.3. Consider a run r in which a user A is sending a message to a user B via the router M. Intuitively, unlinkability of A and B is ensured by showing that there is another run r' that looks similar to r (i.e., is observationally equivalent to r), but in which another user A' is actually communicating with B (and A is communicating with someone else, B'). This situation is illustrated in Figure 3.1. There $((\circ))$ represents a two-layered onion $\{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M$. When peeled the resulting one-layered onion is denoted by $(\circ)$. $((\bullet))$ is a two-layered onion with another nonce in its core.

The following theorem gives a sufficient condition for a run of Onion Routing to provide unlinkability. It says that if there are enough messages in the network, then A and B cannot be linked.

Theorem 3.5.4. *Let r be a run of Onion Routing which contains events*

$$e = (A \to M : \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M)\ \ , e' = (M \to B : \{\!|n|\!\}_B)$$

in this order. Then, there exist an anonymity set $\mathbf{AS}$ *such that* $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B) \wedge \bigwedge_{X\in\mathbf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$*, with* $|\mathbf{AS}| \geq k$*. Recall that k is the size of the router's queue.*

Proof. We first prove the left hand side of the conjunction. That is, we want to prove that $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A,B)$. This means that there should be a run r' with $r' \cong_{\mathsf{spy}} r$ such that $r' \models \neg\varphi_0(A,B)$. Let $e_1 \dots e_q$ be all the messages in r originated by A which have B as final destination, i.e., each e_i has the form $e_i = (A \to M : m_i)$ where $m_i = \{\!|\hat{n}_i, B, \{\!|n_i|\!\}_B|\!\}_M$. To keep the notation easy, we only consider the case where $q = 1$; the general case goes analogously.

By the protocol description of the router, r contains k events $e'_j = (X_j \to M : \{\!|m_j|\!\}_M)$, $j = 1 \dots k$, one of which is e_1. These are the messages that the router M received in the same batch as e_1. Choose one of the events e'_j that is different from e_1 and define a reinterpretation π such that $\pi(m'_j) = m_1$, $\pi(m_1) = m'_j$, extending it in a natural way to the rest of the domain. Now $r' = \pi(r)$ is a run of the protocol (because the router processes messages in batches). Moreover, by the protocol description of the router, the final destinations of e'_j, $j = 1 \dots k$ are pairwise distinct and therefore $B \not\preceq \pi(m_i)$. This implies $r' \not\models \varphi_0(A,B)$.

Now we are going to prove the right side of the conjunction. Given that e' is in r, by the protocol description, there must be k events $e_i = (X_i \to M : m_i)$ with $i = 1 \dots k$ in r. Moreover, one of these events must be e, say e_j. Take $\mathbf{AS} = \{X_i \mid i = 1 \dots k\}$. Note that $|\mathbf{AS}| = k$, since the X_i are all distinct. To show the $r \models \bigwedge_{X \in \mathbf{AS}} \Diamond_{\mathsf{spy}}\varphi_0(X,B)$ part it suffices to show that for a fixed X_i, there is a run $r' = \pi(r)$ such that $r' \models \varphi_0(X_i,B)$. Now take π such that $\pi(m_i) = m_j$, $\pi(m_j) = m_i$, and the obvious extension to the rest of the domain. As before, r' is a run of the protocol and $r' \models \varphi_0(X_i,B)$. $\square$

Finally, let us consider the case where the *private key of the router is compromised*, i.e., $\mathsf{IPo}(\mathsf{spy})$ now contains k_M^{-1}. In this case it is easy to see that unlinkability always fails.

Theorem 3.5.5. *For any run r run of Onion Routing which contains the events*

$$e_1 = (A \to M : \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) \ , \quad e_2 = (M \to B : \{\!|n|\!\}_B)$$

in this order, unlinkability fails. That is, $r \models \Box_{\mathsf{spy}}\varphi_0(A,B)$.

Proof. Let r' be a run such that $r' \cong_{\mathsf{spy}} r$. Say π is a reinterpretation for spy such that $\pi(r) = r'$. Since $k_M^{-1} \in \mathsf{IPo}(\mathsf{spy})$, we have that $\pi(\{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M) = \{\!|n_0, B, \pi(\{\!|n|\!\}_B)|\!\}_M$. Because r' is a valid run of onion routing, $\pi(\{\!|n|\!\}_B) = \{\!|n'|\!\}_B$ for some nonce n'. Then $r' \models A \mathsf{\ Sends\ } n' \wedge B \mathsf{\ Possesses\ } n'$, and therefore $r' \models \varphi_0(A, B)$. □

3.5.3 Onion Routing with subtle flaw

We now propose a version of Onion Routing that has a subtle error, and then show that unlinkability fails while considering a global eavesdropper. As far as we are aware, this is the first framework that detects this kind of subtle flaws.

Imagine a modified version of Onion Routing where each channel in each path is assigned a nonce as an identifier (maybe for establishing a bi-directional communication). These nonces are reused every time that a message is sent through this channel.

The following $\langle e_0, e_1, \ldots, e_7 \rangle$ is an example of a run:

$$\begin{array}{ll} e_0 = (A \to M : \langle n_{11}, ((n)) \rangle) & e_4 = (A \to M : \langle n_{11}, ((n'')) \rangle) \\ e_1 = (B \to M : \langle n_{21}, ((n')) \rangle) & e_5 = (B \to M : \langle n_{31}, ((n''')) \rangle) \\ e_2 = (M \to C : \langle n_{12}, (n) \rangle) & e_6 = (M \to C : \langle n_{12}, (n'') \rangle) \\ e_3 = (M \to C : \langle n_{22}, (n') \rangle) & e_7 = (M \to E : \langle n_{32}, (n''') \rangle) \end{array}$$

where $((-))$ or $(-)$ denotes an onion like in Section 3.5.2, whose ultimate destination is clear from the way it is relayed. In e_2, M generates a nonce (an identifier) n_{12} and remembers the correspondence between n_{11} and n_{12}. When M relays another onion which comes with the same identifier n_{11} (in e_4), it includes the corresponding nonce n_{12} (in e_6). B uses a new identifier n_{31} in e_5 because the destination of the onion is different from e_1. From C's point of view, since the nonces n and n'' come with the same identifier n_{12}, they are from the same originator. Nevertheless the actual originator A is disguised to C.

Notice that in presence of a global eavesdropper, both runs $\langle e_0, \ldots, e_3 \rangle$ and $\langle e_4, \ldots, e_7 \rangle$ ensure unlinkability in themselves. However, when combined, in the run $\langle e_0, \ldots, e_7 \rangle$ the users A and C are linkable, but only

because the combination of identifiers n_{11} and n_{12} occurs twice in the run.

In the following we put this flaw formally. The candidate runs for Flawed Onion Routing should be clear from the above example. A user can originate an onion with an identifier, and the router M relays a peeled onion with a suitable identifier. The identifiers are either freshly generated or reused in the way illustrated in the above example. Again we have users $A, B, C, \ldots$ and a router M which are not corrupted. For simplicity we omit the part where the router sends messages in batches. The protocol description is given by the following rules.

For every user X there are three things he can do

- He can initiate sending a message through the router $(X \to M : \langle n_X, \{\!|n_0, Y, \{\!|n|\!\}_Y|\!\}_M\rangle)$ where n_X, n_0 and n are fresh nonces and $X \neq Y$.
- after such a message, X can now send followup messages by repeating the nonce n_X: $(X \to M : \langle n_X, \{\!|n'_0, Y, \{\!|n'|\!\}_Y|\!\}_M\rangle)$ where n_0 and n are fresh nonces.
- X is still allowed to send padding messages $(X \to M : \langle n_1, \{\!|n_2|\!\}_M\rangle)$ where n_1, n_2 are fresh nonces.

The router M forwards messages, choosing an outgoing nonce n_M corresponding to an incoming nonce n_X. This is, when a message $(X \to M : \langle n_X, \{\!|n_0, Y, m|\!\}_M\rangle)$ is received (again with n_0, m fresh and $X \neq Y$), it sends a message $(M \to Y : \langle n_M, m\rangle)$. The first time that M receives n_X, M picks a fresh random nonce n_M and remember this correspondence for future messages. In general, to attain unlinkability from A to B, there must be another fixed user C who immediately imitates the behavior of A: if A sends an onion bound for B, before the onion is relayed, C must send an onion bound for fixed a D. This condition is fairly unrealistic. The following theorem formally put this (unrealistic) necessary condition for unlinkability, for the case where A sends two onions.

Theorem 3.5.6. *Let r be a run of Flawed Onion Routing which contains*

the events

$$\begin{aligned} e_1 &= (A \to M : \langle n_A, \{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M\rangle), \\ e_2 &= (M \to B : \langle n_M, \{\!|n|\!\}_B\rangle), \\ e_3 &= (A \to M : \langle n_A, \{\!|n'_0, B, \{\!|n'|\!\}_B|\!\}_M\rangle), \\ e_4 &= (M \to B : \langle n_M, \{\!|n'|\!\}_B\rangle) \end{aligned}$$

in this order. If $r \models \neg\Box_{\mathsf{spy}}\varphi_0(A, B)$ *then,* r *contains before* e_2 *and* e_4*, events* $(X \to M : \langle n_X, m_1\rangle)$ *and* $(X \to M : \langle n_X, m_2\rangle)$ *respectively, for some user* $X \neq A$*, messages* m_1, m_2*.*

Proof. Suppose we have a run r such that

$$r \models \neg\Box_{\mathsf{spy}}\varphi_0(A, B),$$

which means that there is a run r' such that $r' \cong_{\mathsf{spy}} r$ and $r' \models \neg\varphi_0(A, B)$. Let π be a reinterpretation for spy such that $\pi(r) = r'$. By Definition 3.3.6, r' must contain events $e'_2 = \pi(e_2) = (M \to B : \langle n_M, \pi(\{\!|n|\!\}_B)\rangle)$ and $e'_4 = \pi(e_4) = (M \to B : \langle n_M, \pi(\{\!|n'|\!\}_B)\rangle)$. Therefore, by the protocol description as to correspondence of identifiers, r' must contain events e'_1 and e'_3 of the form $e'_1 = (X \to M : \langle n_X, \pi(\{\!|n_0, B, \{\!|n|\!\}_B|\!\}_M)\rangle)$ and $e'_3 = (X \to M : \langle n_X, \pi(\{\!|n'_0, B, \{\!|n'|\!\}_B|\!\}_M)\rangle)$, for some common sender X. Now $r' \models \neg\varphi_0(A, B)$ implies $X \neq A$, which finishes the proof. □

The theorem shows that, in order to provide unlinkability, there must be another user in the system that sends repeated nonces to M, with exactly the same pattern as A. This assumption is fairly unrealistic.

3.6 Conclusions and Future Work

We have presented a framework for verifying a variety of information hiding properties, using modal logic. This framework provides a well defined epistemic language where various competing information hiding properties can be expressed and verified uniformly at semantical level. Our fine-grained

definition of message reinterpretation captures subtleties that would not be captured in other state of the art frameworks.

Several extensions of our framework still remain. An obvious one is the consideration of a stronger adversarial model (e.g., active corrupt agents, Dolev-Yao), which should arise straightforwardly. It would also be interesting to consider applications to other fields like secure multiparty computation, and to other security properties besides information hiding, such as secrecy. From a practical perspective, proofs are slightly complicated, even with small examples. Therefore, having a tool which can perform this proofs in an automatic or semi-automatic fashion, would significantly improve the usability of the framework.

Chapter 4

From Formal to Computational Proofs of Protocols using Hashes

4.1 Introduction

The analysis of security protocols is being carried out mainly by means of two different techniques. On the one hand, there is the logic approach, which sees messages as algebraic objects defined using some formal language. In this view, cryptographic operations are symbolic operations which are unbreakable. Attackers are typically modeled as so-called Dolev-Yao attackers [DY83], having total control over the network, having no computational limitations, and being only (but absolutely) incapable of breaking cryptographic operations. This view is appealing, because it is relatively easy to use and captures most mistakes commonly made in security protocols.

On the other hand, there is the complexity-based approach. Here messages are bitstrings and cryptographic operations are functions on bit strings satisfying certain security properties [Yao82, GM84b, Gol01]. Common security notions like secrecy, authenticity, and integrity are formulated

in terms of the probability that someone can mount a successful attack. An attacker here is a resource bounded probabilistic algorithm, limited by running time and/or memory. The complexity based methods are more general and more realistic, but also more complex to use.

In the last few years much research has been done to relate these two perspectives [AR02, AJ01, MW04a, MW04b, Her05]. Such a relation takes the form of a function mapping symbolic messages m to (distributions over) bitstrings $[\![m]\!]$. This map then should relate messages that are observationally equivalent in the symbolic world (meaning that a Dolev-Yao attacker can see no difference between them) to indistinguishable distributions over bitstrings (meaning that a computationally bounded adversary can only with negligible probability distinguish the distributions). Such a map allows one to use formal methods, possibly even automated, to reason about security properties of protocols and have those reasonings be valid also in the computational world.

The work carried out in the literature on relating these two perspectives mainly deals with symmetric encryption [AR02, MW04a] and public key encryption [Her05]. Micciancio and Warinschi [MW04a] briefly but explicitly question if this logical approach can be extended to, among other things, collision resistant hashes. Backes, Pfitzmann, and Waidner [BPW06] show that in their universal composability framework [PW00] a sound interpretation of hashes cannot exist, but that it is possible to give a sound interpretation of formal hashes in the universal composability framework using random oracles. Similar results, also in the random oracle model, have been recently shown by Cortier et al. [CKKW06]. Random oracles are often used in the literature to model hash functions, although they are also often criticized for being unsound in the standard model; there exist examples of protocols that are secure in the random oracle model but provably insecure for every possible instantiation with a real function [CGH04].

The problem with hashes is that in the symbolic world $h(m)$ and $h(m')$ are indistinguishable for a Dolev-Yao attacker if the attacker does not know m or m'. In the computational world, however, the normal security definition — it must be computationally infeasible to compute any pre-image

of a hash value or a hash collision [RS04] — does not guarantee that the hash function hides all partial information about the message; hence there is no guarantee that their interpretation as bitstrings $[\![h(m)]\!]$ and $[\![h(m')]\!]$ are computationally indistinguishable.

A possible solution to this can be found in the work of Canetti and others [Can97a, CMR98] on perfectly one-way functions (a.k.a. oracle hashing). These are computable probabilistic hash functions that hide all partial information of their input (see Section 4.3.3 for the definition and an example).

Our contribution. We propose an extension to the commonly used Abadi-Rogaway logic of symbolic messages introducing a *probabilistic hash operator* $h^r(m)$ in the logic, next to the probabilistic symmetric encryption operator $\{\!|m|\!\}_k^r$. Just as the original logic introduces a $\Box$-operator to put in place of undecryptable ciphertext (for us $\Box^r$, since we also deal with repetitions of ciphertexts), we introduce a $\boxtimes^r$-operator to put in place of the hash of an unknown message. In the computational world, we interpret h as a perfectly one-way function $\mathcal{H}$. We prove that if the encryption algorithm $\mathcal{E}$ is type-0 secure, the resulting interpretation is sound, extending the result from [AR02]. The same result holds if $\mathcal{E}$ is IND-CPA. The case of hashes is of particular interest as the gap between the intuitive concept of a hash function and the computational security assumptions is larger. Moreover, the fact that there is no secret information for the adversary besides the pre-image itself, makes the soundness proof more involved. This result previously appeared, in abbreviated form, as [GvR06b]. Furthermore, assuming that the encryption scheme is confusion-free (i.e., that it is possible to detect decryption failure), we prove that the interpretation is complete as well, extending the result from [MW04a].

For these results, the adversaries under consideration are passive: they do not try to actively modify messages or insert messages into the network. We show that although an oracle hash scheme allows us to port security results from the Dolev-Yao setting to the computational setting with respect to passive adversaries, it does not do so with respect to active adversaries.

The reason is the same as for type-0 or IND-CPA encryption schemes: malleability, i.e., in such an encryption scheme it might be possible for an attacker to modify an encryption $\mathcal{E}(\kappa, \mu)$ of μ to the encryption $\mathcal{E}(\kappa, f(\mu))$ of another message that has some relation to μ; similarly, for a hash function or even an oracle hash scheme it might be possible for an attacker to modify the hash of an unknown message μ to the hash of another unknown message that has some relation to μ. Concretely, we construct an explicit oracle hash scheme in which an adversary can create the hash value $\mathcal{H}(\mu'\mu)$ given only $\mathcal{H}(\mu)$ and μ'. We stress that passive adversaries are the best that one can hope for as the standard security definitions for hash functions or even hash schemes are not resilient against active adversaries.

Overview. Section 4.2 introduces the message algebra, including the probabilistic encryption and probabilistic hash operators. It also defines the observational equivalence relation on messages. Section 4.3 then introduces the computational world, giving the security definitions for encryption and hashes. In Section 4.4 the semantic interpretation $[\![-]\!]$ is defined and Section 4.5 proves the soundness of this interpretation and Section 4.6 completeness. Section 4.7 discusses the limitation to passive adversaries. Finally, Section 4.8 has some concluding remarks.

4.2 The Symbolic Setting

This section describes the message space and the observational equivalence extending the well-known Abadi-Rogaway logic [AR02] of symbolic messages with hashes. These messages are used to describe cryptographic protocols and the observational equivalence tells whether or not two protocol runs are indistinguishable for a global eavesdropper. Here a protocol run is simply the concatenation of all the messages exchanged in the run. In case that the initial possessions of the adversary need to be modeled, they can be concatenated say at the beginning of the message. Note that this manner of modeling protocols is slightly different from the one in the previous chapter. We opt for this model here as it is more convenient at this level

of abstraction.

Definition 4.2.1. **Key** is an infinite set of *key symbols*, **Nonce** an infinite set of *nonce symbols*, **Const** a finite set of *constant symbols*, and **Random** an infinite set of *randomness labels*. Keys are denoted by $k, k', \ldots$, nonces by $n, n', \ldots$, constants by $c, c', \ldots$, and randomness labels by $r, r', \ldots$. There is one special key called $k_\square$ and for every randomness label r there is a special nonce called $n^r_\boxtimes$. Using these building blocks, *messages* are constructed using symbolic encryption, hashing, and pairing operations:

$$\mathbf{Msg} \ni m := c \mid k \mid n \mid \{\!|m|\!\}^r_k \mid h^r(m) \mid \langle m, m\rangle \mid \square^r \mid \boxtimes^r .$$

Here k and n do not range over all keys/nonces, but only over the non-special ones. Special symbols ($\square^r$ and $\boxtimes^r$) are used to indicate undecryptable ciphertexts or hash values of unknown messages. When interpreting messages as (ensembles of distributions over) bitstrings, we will treat $\square^r$ as if it were $\{\!|0|\!\}^r_{k_\square}$ and $\boxtimes^r$ as if it were $h^r(n^r_\boxtimes)$.

A message of the form $\{\!|m|\!\}^r_k$ is called an *encryption* and the set of all such messages is denoted by **Enc**. Similarly, messages of the form $h^r(m)$ are called *hash values* and the set of all these messages is denoted by **Hash**. Finally **Box** denotes the set of all messages of the form $\square^r$ or $\boxtimes^r$. The set of all messages that involve a "random choice" at their "top level", i.e., $\mathbf{Key} \cup \mathbf{Nonce} \cup \mathbf{Enc} \cup \mathbf{Hash} \cup \mathbf{Box}$, is denoted by **RndMsg**. The randomness labels model the fact that our computational encryption scheme is probabilistic: $\langle \{\!|m|\!\}^r_k, \{\!|m|\!\}^r_k \rangle$ models a repetition (forwarding) of the same ciphertext, whereas $\langle \{\!|m|\!\}^r_k, \{\!|m|\!\}^{r'}_k \rangle$ models a re-encryption with the same key. Note that our computational hash scheme, oracle hashing, is also probabilistic; hence, also the symbolic hash function is equipped with randomness labels. In the computational setting there is a verification function that tells whether or not a certain hash value corresponds to a certain message.

For the sake of readability we omit angle brackets inside encryptions and hashes and anywhere else when redundant.

The *closure* of a set U of messages is the set of all messages that can be constructed from U using tupling, detupling, and decryption. It represents

the information an adversary could deduce knowing U. Note that, due to the (perfect) one-wayness of a hash function, knowing $h^r(m)$ does not provide an adversary with any information about m. Similarly, modeling the randomization of the encryption and hash schemes, building new encryptions or hash values does not give any useful information to the adversary.

Definition 4.2.2 (Closure)**.** Let U be a set of messages. The *closure* of U, denoted by $\overline{U}$, is the smallest set of messages satisfying:

1. $\mathbf{Const} \subseteq \overline{U}$;
2. $U \subseteq \overline{U}$;
3. $m, m' \in \overline{U} \implies \langle m, m' \rangle \in \overline{U}$;
4. $\{\!|m|\!\}_k^r, k \in \overline{U} \implies m \in \overline{U}$;
5. $\langle m, m' \rangle \in \overline{U} \implies m, m' \in \overline{U}$.

For the singleton set $\{m\}$, we write $\overline{m}$ instead of $\overline{\{m\}}$.

The function $\mathsf{pattern}\colon \mathbf{Msg} \to \mathbf{Msg}$ is a straightforward extension of the same function in Abadi-Rogaway [AR02] which takes a message m and reduces it to a pattern. Intuitively, $\mathsf{pattern}(m)$ is the pattern that an attacker sees in a message m and messages with the same pattern (up to renaming) look the same to her. Encryptions with keys the attacker cannot learn (i.e., not in $\overline{m}$) are replaced by $\square$ and hash values of unknown messages by $\boxtimes$. Since we allow repetition of ciphertexts and hash values, we have to distinguish between unknown, but equal, ciphertexts $\langle \square^r, \square^r \rangle$ and unknown, but distinct, ciphertexts $\langle \square^r, \square^{r'} \rangle$ and likewise for hashes.

Definition 4.2.3. The function $\mathsf{pattern}\colon \mathbf{Msg} \to \mathbf{Msg}$ is defined by

$$\mathsf{pattern}(m) = \mathsf{pattern}(m, \overline{m})$$

where

$$\begin{aligned}
\mathsf{pattern}(\langle m_1, m_2\rangle, U) &= \langle \mathsf{pattern}(m_1, U), \mathsf{pattern}(m_2, U)\rangle \\
\mathsf{pattern}(\{\!|m|\!\}_k^r, U) &= \begin{cases} \{\!|\mathsf{pattern}(m, U)|\!\}_k^r, & \text{if } k \in U; \\ \square^{\mathcal{R}(\{\!|m|\!\}_k^r)}, & \text{otherwise.} \end{cases} \\
\mathsf{pattern}(h^r(m), U) &= \begin{cases} h^r(\mathsf{pattern}(m, U)), & \text{if } m \in U; \\ \boxtimes^{\mathcal{R}(h^r(m))}, & \text{otherwise.} \end{cases} \\
\mathsf{pattern}(m, U) &= m \quad \text{in any other case.}
\end{aligned}$$

Here $\mathcal{R}\colon \mathbf{Enc} \cup \mathbf{Hash} \hookrightarrow \mathbf{Random}$ is an injective function that takes an encryption or a hash value and outputs a tag that identifies its randomness.

The tagging function $\mathcal{R}$ is needed to make sure that the function $\mathsf{pattern}$ is injective: distinct undecryptable messages should be replaced by distinct boxes and similarly for hashes. Note that instead of using $\mathcal{R}$ one could also tacitly assume that randomness labels r used in distinct contexts, such as $\{\!|m|\!\}_k^r$ and $\{\!|m'|\!\}_k^{r'}$ with $m \neq m'$, are always distinct.

Example 4.2.4. Consider the message

$$m = \langle \{\!|\{\!|1|\!\}_{k'}^{r'}, h^{\tilde{r}}(n)|\!\}_k^r, h^{\hat{r}}(k), k\rangle.$$

Then

$$\mathsf{pattern}(m) = \langle \{\!|\ \square^s\ ,\ \boxtimes^t\ |\!\}_k^r, h^{\hat{r}}(k), k\rangle, \quad \text{because } k', n \text{ are not in } \overline{m},$$

where $t = \mathcal{R}(h^{\tilde{r}}(n))$ and $s = \mathcal{R}(\{\!|1|\!\}_{k'}^{r'})$.

Definition 4.2.5 (Renaming)**.** Two messages m and m' are said to be *equivalent up to renaming*, notation $m \approx m'$, if there is a type preserving permutation σ of $\mathbf{Key} \cup \mathbf{Nonce} \cup \mathbf{Box} \cup \mathbf{Random}$ such that $m = m'\sigma$. Here $m'\sigma$ denotes simultaneous substitution of x by $\sigma(x)$ in m', for all $x \in \mathbf{Key} \cup \mathbf{Nonce} \cup \mathbf{Box} \cup \mathbf{Random}$.

Definition 4.2.6 (Observational equivalence)**.** Two messages m and m' are said to be *observationally equivalent*, denoted by $m \cong m'$, if $\mathsf{pattern}(m) \approx \mathsf{pattern}(m')$.

In the computational world, security definitions for encryption schemes do not guarantee security when encryption cycles occur. In the symbolic world, however, this poses no problem. Therefore, as in the original setting in [AR02], for the mapping from the symbolic world to the computational world to be sound, we have to forbid encryption cycles in the symbolic world.

Definition 4.2.7 (Sub-message)**.**

1. $m_1 \preceq \langle m_1, m_2 \rangle$
2. $m_2 \preceq \langle m_1, m_2 \rangle$
3. $m \preceq \{\!|m|\!\}_k^r$
4. $m \preceq h^r(m)$.

Definition 4.2.8 (Acyclicity)**.** Let m be a message and k, k' two keys. The key k is said to *encrypt* k' *in* m if m has a sub-message of the form $\{\!|m'|\!\}_k^r$ with k' being a sub-message of m'. A message is said to be *acyclic* if there is no sequence $k_1, k_2, \ldots, k_n, k_{n+1} = k_1$ of keys such that k_i encrypts k_{i+1} in m for all $i \in \{1, \ldots, n\}$.

4.3 The Computational Setting

This section gives a brief overview of some of the concepts used in the complexity theoretic approach to security protocols. Much of this is standard; the reader is referred to [GB01, BDJR97] for a thorough treatment of the basic concepts, to [AR02] for the notion of *type-0 security* for cryptographic schemes (see Section 4.3.2 below), and to [Can97a] for the notion of *oracle hashing* (see Section 4.3.3 below).

In the computational world, messages are elements of $\mathbf{Str} := \{0,1\}^*$. Cryptographic algorithms and adversaries are probabilistic polynomial-time algorithms. When analyzing cryptographic primitives, it is customary to consider probabilistic algorithms that take an element in $\mathbf{Param} := \{1\}^*$ as input, whose length scales with the security parameter. By making the security parameter large enough, the system should become arbitrarily hard to break.

This idea is formalized in the security notions of the cryptographic operations. The basic one, which is what is used to define the notion of semantically equivalent messages, is that of *computational indistinguishability* of probability ensembles over **Str**. Here a *probability ensemble over* **Str** is a sequence $\{A_\eta\}_{\eta\in\mathbb{N}}$ of probability distributions over **Str** indexed by the security parameter.

Definition 4.3.1 (Computational indistinguishability)**.** Two probability ensembles $\{X_\eta\}_{\eta\in\mathbb{N}}$ and $\{Y_\eta\}_{\eta\in\mathbb{N}}$ are *computationally indistinguishable*, notation $\{X_\eta\}_{\eta\in\mathbb{N}} \equiv \{Y_\eta\}_{\eta\in\mathbb{N}}$, if for every probabilistic polynomial-time algorithm D,

$$\mathbb{P}[x \leftarrow X_\eta; D(1^\eta, x) = 1] - \mathbb{P}[x \leftarrow Y_\eta; D(1^\eta, x) = 1]$$

is a negligible function of η.

Recall that a function $f\colon \mathbb{N} \to \mathbb{N}$ is called *negligible* if for all positive polynomials p, $f(\eta) \leq \frac{1}{p(\eta)}$ for large enough η. After a brief interlude on probabilistic polynomial-time algorithms in Section 4.3.1, we give the formal definition of an encryption scheme and its security notion in Section 4.3.2 and of oracle hashing in Section 4.3.3.

4.3.1 Probabilistic Algorithms

In Definition 4.3.1, the notion of probabilistic polynomial-time algorithm was already used. Because we explicitly use two different views of these algorithms and in order to fix notation, we give a more precise definition.

Definition 4.3.2. Coins is the set $\{0,1\}^\omega$, the set of all infinite sequences of 0's and 1's. We equip Coins with the probability distribution obtained by flipping a fair coin for each element in the sequence.

Definition 4.3.3. The result of running a probabilistic algorithm A on an input $x \in$ **Str** is a probability distribution $A(x)$ over **Str**. When we need to explicitly write the randomness used while running A, we write $A(x,\rho)$ with $\rho \in$ Coins. Using this notation, $\mathbb{P}[A(x) = y] = \mathbb{P}[\rho \leftarrow \mathsf{Coins}; A(x,\rho) =$

$y]$. When confusion is unlikely, we will also denote the support of this probability distribution, $\{y \in \mathbf{Str} \mid \mathbb{P}[\rho \leftarrow \mathsf{Coins}; A(x,\rho) = y)] > 0\}$, by $A(x)$.

Now suppose that A runs in polynomial time p. Then running A on x cannot use more than $p(|x|)$ coin flips. Letting $\mathsf{Coins}_{p(|x|)}$ denote the uniform probability distribution on $\{0,1\}^{p(|x|)}$, we can also write $\mathbb{P}[A(x) = y] = \mathbb{P}[\rho \leftarrow \mathsf{Coins}_{p(|x|)}; A(x,\rho) = y]$.

Definition 4.3.4. A probabilistic polynomial time algorithm $P : \mathsf{Dom}(P) \to \{0,1\}$ is called a probabilistic polynomial time predicate.

4.3.2 Encryption Scheme

For each security parameter $\eta \in \mathbb{N}$ we let $\mathbf{Plaintext}_\eta \subseteq \mathbf{Str}$ be a non-empty set of *plaintexts*, satisfying that for each $\eta \in \mathbb{N}$: $\mathbf{Plaintext}_\eta \subseteq \mathbf{Plaintext}_{\eta+1}$ as in Goldwasser and Bellare [GB01]. Let us define $\mathbf{Plaintext} = \bigcup_\eta \mathbf{Plaintext}_\eta$. There is a set $\mathbf{Keys} \subseteq \mathbf{Str}$ of *keys* and also a set $\mathbf{Ciphertext} \subseteq \mathbf{Str}$ of *ciphertexts*. Furthermore, there is a special bitstring $\perp$ not appearing in **Plaintext** or **Ciphertext**. An *encryption scheme* Π consists of three algorithms:

1. a (probabilistic) key generation algorithm $\mathcal{K}$: $\mathbf{Param} \to \mathbf{Keys}$ that outputs, given a unary sequence of length η, a randomly chosen element of **Keys**;
2. a (probabilistic) encryption algorithm $\mathcal{E}$: $\mathbf{Keys} \times \mathbf{Str} \to \mathbf{Ciphertext} \cup \{\perp\}$ that outputs, given a key and a bitstring, an element from **Ciphertext** or $\perp$;
3. a (deterministic) decryption algorithm $\mathcal{D}$: $\mathbf{Keys} \times \mathbf{Str} \to \mathbf{Plaintext} \cup \{\perp\}$ that outputs, given a key and a ciphertext, an element from **Plaintext** or $\perp$.

These algorithms must satisfy that the decryption (with the correct key) of a ciphertext returns the original plaintext. The element $\perp$ is used to indicate failure of en- or decryption, although, beforehand, there is no requirement that decrypting with the wrong keys yields $\perp$. Now we define

type-0 security of an encryption scheme as in [AR02], which is a variant of the standard semantic security definition, enhanced with some extra properties. In particular a type-0 secure encryption scheme is which-key concealing, repetition concealing and length hiding. We refer to the original paper for motivation and explanations on how to achieve such an encryption scheme. The notion of type-0 security makes slightly unrealistic assumptions on the encryption scheme. However our result on hashes does not significantly depend on the specific security notion for the encryption scheme. As in [MP05, Her05], it is possible to replace type-0 security by the standard notion of IND-CPA or IND-CCA by adapting the definition of pattern. For simplicity of the exposition, throughout this chapter we adopt the former security notion.

Definition 4.3.5. An *adversary (for type-0 security)* is a probabilistic polynomial-time algorithm $A^{\mathcal{F}(-),\mathcal{G}(-)} \colon \mathbf{Param} \to \{0,1\}$ having access to two probabilistic oracles $\mathcal{F}, \mathcal{G} \colon \mathbf{Str} \to \mathbf{Str}$. The *advantage* of such an adversary is the function $\mathsf{Adv}_A \colon \mathbb{N} \to \mathbb{R}$ defined by

$$\mathsf{Adv}_A(\eta) = \mathbb{P}[\kappa, \kappa' \leftarrow \mathcal{K}(1^\eta); A^{\mathcal{E}(\kappa,-),\mathcal{E}(\kappa',-)}(1^\eta) = 1] - \\ \mathbb{P}[\kappa \leftarrow \mathcal{K}(1^\eta); A^{\mathcal{E}(\kappa,0),\mathcal{E}(\kappa,0)}(1^\eta) = 1].$$

Here the probabilities are taken over the choice of κ and κ' by the key generation algorithm $\mathcal{K}$, over the internal coins used by the oracles, and over the internal choices of A. An encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is called *type-0 secure* if for all polynomial-time adversaries A as above, the advantage Adv_A is a negligible function of η.

In the sequel we need an extra assumption on the encryption scheme, namely that the ciphertexts are well-spread as a function of the coins tosses of $\mathcal{E}$. It means that for *all* plaintexts μ and *all* keys κ, no ciphertext is exceptionally likely to occur as the encryption of μ under κ. Note that this does not follow from, nor implies type-0 security. Also note that every encryption scheme running in cipher block chaining mode automatically has this property: the initial vector provides the required randomness.

Definition 4.3.6 (Well-spread)**.** An encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is said to be *well-spread* if

$$\sup_{x \in \mathbf{Ciphertext}, \kappa \in \mathcal{K}(1^\eta), \mu \in \mathbf{Plaintext}_\eta} \mathbb{P}[\mathcal{E}(\kappa, \mu) = x]$$

is a negligible function of η.

For completeness it is needed that the decryption algorithm returns *reject* whenever it is called with a key that was not used to encrypt the message in the first place. The special bitstring $\perp$ is used to indicate failure of decryption. This property is called *confusion freeness*. See [MW04a], where the completeness for the original Abadi-Rogaway logic is proven.

Definition 4.3.7 (Confusion freeness)**.** Let $\Pi = \langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ be an encryption scheme indexed by the security parameter η. Π is said to be *confusion free* if for all bitstrings μ the probability

$$\mathbb{P}[\kappa_1, \kappa_2 \leftarrow \mathcal{K}(\eta) : \mathcal{D}_{\kappa_1}(\mathcal{E}_{\kappa_2}(\mu)) \neq \perp]$$

is a negligible function of η.

4.3.3 Oracle Hashing

The underlying secrecy assumptions behind formal or Dolev-Yao hashes [DY83] are very strong. It is assumed that given a hash value $f(x)$, it is not possible for an adversary to learn any information about the pre-image x. In the literature this idealization is often modelled with the random oracle [BR93]. Such a primitive is not computable and therefore it is also an idealization. Practical hash functions like SHA or MD5 are very useful cryptographic primitives but have no proven security guarantees. Moreover, under the traditional security notions (one-wayness), a function that reveals half of its input might still be secure. In addition, any deterministic hash function f leaks partial information about x, namely $f(x)$ itself. Throughout this chapter we consider a new primitive introduced by Canetti [Can97a] called *oracle hashing*, that mimics what semantic security is for encryption

schemes. This hash function is probabilistic and therefore it needs a verification function, just as in a signature scheme. A *hash scheme* consists of two algorithms $\mathcal{H}$ and $\mathcal{V}$. The probabilistic algorithm $\mathcal{H}\colon \mathbf{Param} \times \mathbf{Str} \to \mathbf{Str}$ takes a unary sequence and a message and outputs a hash value; the verification algorithm $\mathcal{V}\colon \mathbf{Str} \times \mathbf{Str} \to \{0,1\}$ that given two messages x and c correctly decides whether c is a hash of x or not. As an example we reproduce here a hash scheme proposed in the original paper. Let p be a large (i.e., scaling with η) safe prime. Take $\mathcal{H}(x) = \langle r^2 \mod p, r^{2 \cdot f(x)} \mod p \rangle$, where r is a randomly chosen element in $\mathbb{Z}_p^*$ and f is any collision resistant hash function. The verification algorithm $\mathcal{V}(x, \langle a, b \rangle)$ just checks whether $b = a^{f(x)} \mod p$.

We consider two security notions for such a hash scheme. The first one, proposed by Canetti [Can97a] and later revisited in [CMR98], *oracle indistinguishability*, guarantees that an adversary can gain no information at all about a bitstring, given its hash value (or rather, with sufficiently small probability). The second one, more standard, is an appropriate form of collision resistance. It guarantees that an adversary cannot (or rather, again, with sufficiently small probability) compute two distinct messages that successfully pass the verification test with the same hash value.

Definition 4.3.8 (Oracle indistinguishability)**.** A hash scheme $\langle \mathcal{H}, \mathcal{V} \rangle$ is said to be *oracle indistinguishable* if for every family of probabilistic polynomial-time predicates $\{D_\eta\colon \mathbf{Str} \to \{0,1\}\}_{\eta \in \mathbb{N}}$ and every positive polynomial p there is a polynomial size family $\{L_\eta\}_{\eta \in \mathbb{N}}$ of subsets of $\mathbf{Str}$ such that for all large enough η and all $x, y \in \mathbf{Str} \setminus L_\eta$:

$$\mathbb{P}[D_\eta(\mathcal{H}(1^\eta, x)) = 1] - \mathbb{P}[D_\eta(\mathcal{H}(1^\eta, y)) = 1] < \frac{1}{p(\eta)}.$$

Here the probabilities are taken over the choices made by $\mathcal{H}$ and the choices made by D_η. This definition is the non-uniform [Gol01] version of oracle indistinguishability proposed by Canetti [Can97a] as it is used in the proofs in Appendix B of the full version [Can97b].

Definition 4.3.9 (Collision resistance)**.** A hash scheme $\langle \mathcal{H}, \mathcal{V} \rangle$ is said to be *collision resistant* if for every probabilistic polynomial time adversary

A, the probability

$$\mathbb{P}[\langle c, x, y\rangle \leftarrow A(1^\eta); x \neq y \wedge \mathcal{V}(x, c) = \mathcal{V}(y, c) = 1]$$

is a negligible function of η.

4.4 Interpretation

Section 4.2 describes a setting where messages are symbolic terms generated by some grammar. In Section 4.3 messages are bitstrings and operations are given by probabilistic algorithms operating on bitstrings. This section shows how to map symbolic messages to (distributions over) bitstrings. This interpretation is very much standard. We refer to [AR02, AJ01, MW04a] for a thorough explanation. In particular this section introduces notation that allows us to assign, beforehand, some of the random coin flips used for the computation of the interpretation of a message. This notation becomes useful throughout the soundness proof.

Tagged representation. Throughout this chapter we assume that it is always possible to recover the type information of a message from its bitstring representation. This can be easily achieved by adding the necessary type tags to the bitstring representation. We will abstract from this representation by overloading the notation. We use Greek letters for bitstrings and μ represents a bitstring of a generic type. We write $\mu_1\mu_2$ for a pair of bitstrings (in [AR02] this would be written as $\langle(\mu_1, \mu_2), \text{“pair”}\rangle$); ϵ for a ciphertext; κ for a key; ψ for a hash value; ν for a nonce and ς for a constant.

Definition 4.4.1. For every message m we define the set $\mathsf{R}(m) \subseteq \mathbf{Msg}$ of *random messages in* m as follows:

$$\begin{aligned}
\mathsf{R}(c) &= \emptyset & \mathsf{R}(\{\!|m|\!\}_k^r) &= \mathsf{R}(m) \cup \{k, \{\!|m|\!\}_k^r\}\\
\mathsf{R}(n) &= \{n\} & \mathsf{R}(h^r(m)) &= \mathsf{R}(m) \cup \{h^r(m)\}\\
\mathsf{R}(k) &= \{k\} & \mathsf{R}(\langle m_1, m_2\rangle) &= \mathsf{R}(m_1) \cup \mathsf{R}(m_2)\\
\mathsf{R}(\square^r) &= \{k_\square, \square^r\} & \mathsf{R}(\boxtimes^r) &= \{n^r_\boxtimes, \boxtimes^r\}.
\end{aligned}$$

Note that $\mathsf{R}(m)$ is nearly equal to the set of all sub-messages of m that are in **RndMsg** (defined in Section 4.2); the only difference is that $\mathsf{R}(m)$ also may contain the special key $k_\square$ or special nonces $n^r_\boxtimes$. When interpreting a message m as (ensembles of distributions over) bitstrings (Definition 4.4.3 below), we will first choose a sequence of coin flips for all elements of $\mathsf{R}(m)$ and use these sequences as source of randomness for the appropriate interpretation algorithms.

Definition 4.4.2. For every finite subset X of **RndMsg** we define $\mathsf{Coins}(X)$ as $\{\tau \mid \tau\colon X \to \mathsf{Coins}\}$. We equip $\mathsf{Coins}(X)$ with the induced product probability distribution. Furthermore, for every message m we write $\mathsf{Coins}(m)$ instead of $\mathsf{Coins}(\mathsf{R}(m))$.

An element of τ of $\mathsf{Coins}(m)$ gives, for every sub-message m' of m that requires random choices when interpreting this sub-message as a bitstring, an infinite sequence $\tau(m')$ of coin flips that will be used to resolve the randomness.

Now we are ready to give semantic to our message algebra. We use $\mathcal{E}$ (as defined in Section 4.3.2) to interpret encryptions, $\mathcal{K}$ to interpret key symbols, and $\mathcal{H}$ (as defined in Section 4.3.3) to interpret hashes. We let $\mathcal{C}\colon \mathbf{Const} \to \mathbf{Str}$ be a function that (deterministically) assigns a constant bit string to each constant identifier. We let $\mathcal{N}\colon \mathbf{Param} \to \mathbf{Str}$ be the nonce generation function that, given a unary sequence of length η, chooses uniformly and randomly a bitstring from $\{0,1\}^\eta$.

Definition 4.4.3. For a message m, a value of the security parameter $\eta \in \mathbb{N}$, a finite set U of messages containing $\mathsf{R}(m)$, and a $\tau \in \mathsf{Coins}(U)$, we can (deterministically) create a bitstring $[\![m]\!]^\tau_\eta \in \mathbf{Str}$ as follows:

$$
\begin{aligned}
[\![c]\!]^\tau_\eta &= \mathcal{C}(c) & [\![\{\!|m|\!\}^r_k]\!]^\tau_\eta &= \mathcal{E}([\![k]\!]^\tau_\eta, [\![m]\!]^\tau_\eta, \tau(\{\!|m|\!\}^r_k))\\
[\![k]\!]^\tau_\eta &= \mathcal{K}(1^\eta, \tau(k)) & [\![h^r(m)]\!]^\tau_\eta &= \mathcal{H}(1^\eta, [\![m]\!]^\tau_\eta, \tau(h^r(m)))\\
[\![n]\!]^\tau_\eta &= \mathcal{N}(1^\eta, \tau(n)) & [\![\square^r]\!]^\tau_\eta &= \mathcal{E}([\![k_\square]\!]^\tau_\eta, \mathcal{C}(0), \tau(\square^r))\\
[\![\langle m_1, m_2\rangle]\!]^\tau_\eta &= [\![m_1]\!]^\tau_\eta [\![m_2]\!]^\tau_\eta & [\![\boxtimes^r]\!]^\tau_\eta &= \mathcal{H}(1^\eta, [\![n^r_\boxtimes]\!]^\tau_\eta, \tau(\boxtimes^r)).
\end{aligned}
$$

Note that $[\![m]\!]_\eta^\tau = [\![m]\!]_\eta^{\tau|_{\mathsf{R}(m)}}$. For a fixed message m and $\eta \in \mathbb{N}$, choosing τ from the probability distribution $\mathsf{Coins}(\mathsf{R}(m))$ creates a probability distribution $[\![m]\!]_\eta$ over **Str**:

$$[\![m]\!]_\eta := [\tau \leftarrow \mathsf{Coins}(m); [\![m]\!]_\eta^\tau].$$

Note that although the codomain of $\tau \in \mathsf{Coins}(m)$ is Coins, the set of *infinite* bitstrings, when interpreting a fixed message m at a fixed value of the security parameter η, only a predetermined *finite* initial segment of each sequence of coin flips will be used by $\mathcal{K}$, $\mathcal{N}$, $\mathcal{E}$, and $\mathcal{H}$ (cf. Definition 4.3.3). In fact, because these four algorithms run in polynomial time, for every message m there exists a polynomial p_m such that every call to one of these four algorithms uses at most the first $p_m(\eta)$ elements of τ. Now define $\mathsf{Coins}_\eta(m) = \{\tau \mid \tau \colon \mathsf{R}(m) \to \{0,1\}^{p_m(\eta)}\}$ and equip this with the uniform probability distribution. Then we can also write

$$[\![m]\!]_\eta = [\tau \leftarrow \mathsf{Coins}_\eta(m); [\![m]\!]_\eta^\tau].$$

Furthermore, letting η range over $\mathbb{N}$ creates an ensemble of probability distributions $[\![m]\!]$ over **Str**, namely $[\![m]\!] := \{[\![m]\!]_\eta\}_{\eta\in\mathbb{N}}$.

4.4.1 Partial interpretation

For technical reasons throughout the soundness proof, we need to compute the interpretation of a symbolic message while part of its randomness has already been chosen. This section introduces straightforward notation to do so.

Definition 4.4.4. For every message m and m' we define the set $\mathsf{R}(m, m') \subseteq$ **RndMsg** of *random messages in m relative to m'* as follows:
if $m = m'$, then $\mathsf{R}(m, m') = \emptyset$, otherwise

$$\begin{aligned}
\mathsf{R}(c, m') &= \emptyset & \mathsf{R}(\{\!|m|\!\}_k^r, m') &= \mathsf{R}(m, m') \cup \{k, \{\!|m|\!\}_k^r\} \\
\mathsf{R}(n, m') &= \{n\} & \mathsf{R}(h^r(m), m') &= \mathsf{R}(m, m') \cup \{h^r(m)\} \\
\mathsf{R}(k, m') &= \{k\} & \mathsf{R}(\langle m_1, m_2\rangle, m') &= \mathsf{R}(m_1, m') \cup \mathsf{R}(m_2, m') \\
\mathsf{R}(\square^r, m') &= \{k_\square, \square^r\} & \mathsf{R}(\boxtimes^r, m') &= \{n^r_\boxtimes, \boxtimes^r\}.
\end{aligned}$$

Note that $\mathsf{R}(m, m')$ is the set of all random messages in m except those that *only* occur as a sub-message of m'.

Example 4.4.5. Let m be the message $\langle k, \{\!|0|\!\}^r_k, h^{r'}(\{\!|0|\!\}^r_k, n), n'\rangle$ and let $\tilde{m}$ be the message inside the hash: $\langle \{\!|0|\!\}^r_k, n\rangle$. Then the randomness in m is $\mathsf{R}(m) = \{k, \{\!|0|\!\}^r_k, h^{r'}(\{\!|0|\!\}^r_k, n), n', n\}$, the randomness inside the hash is $\mathsf{R}(\tilde{m}) = \{\{\!|0|\!\}^r_k, k, n\}$, and the randomness that occurs only outside the hash is $\mathsf{R}(m, h^{r'}(\tilde{m})) = \mathsf{R}(m) \setminus \{h^{r'}(\tilde{m}), n\}$. The randomness that is shared between the inside of the hash and the outside of the hash is $\mathsf{R}(m, h^{r'}(\tilde{m})) \cap \mathsf{R}(\tilde{m}) = \{k, \{\!|0|\!\}^r_k\}$.

We will need a way of interpreting a message as a bitstring when the interpretation of certain sub-messages has already been chosen in some other way.

Definition 4.4.6. Let $\eta \in \mathbb{N}$, let e be a function from $\mathsf{Dom}(e) \subseteq \mathbf{Msg}$ to $\mathbf{Str}$ and let $\tau \in \mathsf{Coins}_\eta(U \setminus \mathsf{Dom}(e))$ with U a finite set of messages containing $\mathsf{R}(m)$. We interpret a message m using e whenever possible. Otherwise, we use the coin flips assigned by τ to generate an interpretation. If $m \in \mathsf{Dom}(e)$, then $[\![m]\!]^{e,\tau}_\eta = e(m)$, else

$$
\begin{aligned}
[\![c]\!]^{e,\tau}_\eta &= \mathcal{C}(c) & [\![\{\!|m|\!\}^r_k]\!]^{e,\tau}_\eta &= \mathcal{E}([\![k]\!]^\tau_\eta, [\![m]\!]^{e,\tau}_\eta, \tau(\{\!|m|\!\}^r_k)) \\
[\![k]\!]^{e,\tau}_\eta &= \mathcal{K}(1^\eta, \tau(k)) & [\![h^r(m)]\!]^{e,\tau}_\eta &= \mathcal{H}(1^\eta, [\![m]\!]^{e,\tau}_\eta, \tau(h^r(m))) \\
[\![n]\!]^{e,\tau}_\eta &= \mathcal{N}(1^\eta, \tau(n)) & [\![\square^r]\!]^{e,\tau}_\eta &= \mathcal{E}([\![k_\square]\!]^{e,\tau}_\eta, \mathcal{C}(0), \tau(\square^r)) \\
[\![\langle m_1, m_2\rangle]\!]^{e,\tau}_\eta &= [\![m_1]\!]^{e,\tau}_\eta [\![m_2]\!]^{e,\tau}_\eta & [\![\boxtimes^r]\!]^{e,\tau}_\eta &= \mathcal{H}(1^\eta, [\![n^r_\boxtimes]\!]^{e,\tau}_\eta, \tau(\boxtimes^r)).
\end{aligned}
$$

We also need a way of pre-specifying some of the random choices to be made when interpreting a message.

Definition 4.4.7. Let $\eta \in \mathbb{N}$ and let $\tau \in \mathsf{Coins}_\eta(U)$ for some finite set of messages U. Then for every message m, the distribution $[\![m]\!]^\tau_\eta$ is obtained by randomly choosing coins for the remaining randomness labels in m. Formally,

$$[\![m]\!]^\tau_\eta := [\tau' \leftarrow \mathsf{Coins}_\eta(\mathsf{R}(m) \setminus U); [\![m]\!]^{\tau \cup \tau'}_\eta],$$

where $\tau \cup \tau' \in \mathsf{Coins}_\eta(m)$ denotes the function which agrees with τ on $U \cap \mathsf{R}(m)$ and with τ' on $\mathsf{R}(m) \setminus U$.

This can also be combined with the previous way of preselecting a part of the interpretation. For a function e from a set $\mathsf{Dom}(e) \subseteq \mathbf{Msg}$ to $\mathbf{Str}$ and $\tau \in \mathsf{Coins}_\eta(U)$ as above, we define

$$[\![m]\!]_\eta^{e,\tau} := [\tau' \leftarrow \mathsf{Coins}_\eta(\mathsf{R}(m) \setminus U); [\![m]\!]_\eta^{e,\tau\cup\tau'}].$$

4.5 Soundness

This section shows that the interpretation proposed in the previous section is computationally sound. Throughout this section we assume that the encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D}\rangle$ is type-0 secure (or IND-CCA with pattern modified as in [Her05, MP05]) and well-spread, and that the probabilistic hash scheme $\langle \mathcal{H}, \mathcal{V}\rangle$ is oracle indistinguishable and collision resistant.

In order to isolate our contribution, we split the function pattern in two parts: one replacing the encryptions and one replacing the hashes. We define the function encpat as in Abadi-Rogaway [AR02] which takes a message m and reduces it to a pattern. This function does not replace hashes.

Definition 4.5.1. The function $\mathsf{encpat}\colon \mathbf{Msg} \to \mathbf{Msg}$ is defined as follows

$$\mathsf{encpat}(m) = \mathsf{encpat}(m, \overline{m})$$

where

$$\begin{aligned}
\mathsf{encpat}(\langle m_1, m_2\rangle, U) &= \langle \mathsf{encpat}(m_1, U), \mathsf{encpat}(m_2, U)\rangle \\
\mathsf{encpat}(\{\!|m|\!\}_k^r, U) &= \begin{cases} \{\!|\mathsf{encpat}(m, U)|\!\}_k^r, & \text{if } k \in U; \\ \square^{\mathcal{R}(\{\!|m|\!\}_k^r)}, & \text{otherwise.} \end{cases} \\
\mathsf{encpat}(h^r(m), U) &= h^r(\mathsf{encpat}(m, U)) \\
\mathsf{encpat}(m, U) &= m \quad \text{in any other case.}
\end{aligned}$$

Now we define the function hashpat which takes a message m and reduces all hashes of unknown (not in $\overline{m}$) sub-messages, to $\boxtimes$. This function does not replace encryptions.

Definition 4.5.2. Define the function $\mathsf{hashpat}\colon \mathbf{Msg} \to \mathbf{Msg}$ as

$$\mathsf{hashpat}(m) = \mathsf{hashpat}(m, \overline{m})$$

where

$$\begin{aligned}
\mathsf{hashpat}(\langle m_1, m_2\rangle, U) &= \langle \mathsf{hashpat}(m_1, U), \mathsf{hashpat}(m_2, U)\rangle \\
\mathsf{hashpat}(\{\!|m|\!\}_k^r, U) &= \{\!|\mathsf{hashpat}(m, U)|\!\}_k^r \\
\mathsf{hashpat}(h^r(m), U) &= \begin{cases} h^r(\mathsf{hashpat}(m, U)), & \text{if } m \in U; \\ \boxtimes^{\mathcal{R}(h^r(m))}, & \text{otherwise.} \end{cases} \\
\mathsf{hashpat}(m, U) &= m \quad \text{in any other case.}
\end{aligned}$$

Lemma 4.5.3. $\mathsf{pattern} \approx \mathsf{encpat} \circ \mathsf{hashpat}$.

Proof. A straightforward induction on m shows that $\overline{m} = \overline{\mathsf{hashpat}(m)}$. Using this, another straightforward induction on m shows that $\mathsf{pattern}(m) \approx \mathsf{encpat}(\mathsf{hashpat}(m))$. □

Theorem 4.5.4 (Abadi-Rogaway)**.** Let m be an acyclic message. Suppose that for every sub-message $h^r(\tilde{m})$ of m, $\tilde{m} \in \overline{m}$. Then $[\![m]\!] \equiv [\![\mathsf{encpat}(m)]\!]$.

Proof. The proof follows just like in Abadi-Rogaway [AR02]. Interpreting hashes here is straightforward because their argument is always known, by assumption. We refer the reader to the original paper for a full proof. □

Before starting the proof of the soundness theorem, we give a sketch for an easy case and point out where the technical problems in the general case are. Consider two messages $h(n, 0)n'$ and $h(n, 1)n'$. These are observationally equivalent and we have to prove that $[\![h(n, 0), n']\!]$ and $[\![h(n, 1), n']\!]$ are computationally indistinguishable. The way to prove this is standard: we assume that there exists a probabilistic polynomial-time adversary A that can distinguish these two ensembles of probability distribution and use it to build an adversary D that break the oracle indistinguishability of the hash scheme $\langle \mathcal{H}, \mathcal{V}\rangle$. What D has to do is clear: it receives a hash value α which is either $\mathcal{H}(1^\eta, \nu 0)$ or $\mathcal{H}(1^\eta, \nu 1)$, and has to guess which one it is. It generates a nonce ν' and calls A on $\alpha\nu'$. Since A can successfully

distinguish $\mathcal{H}(1^\eta, \nu 0)\nu'$ from $\mathcal{H}(1^\eta, \nu 1)\nu'$, this enables D to break oracle indistinguishability.

A problem occurs in the case of the messages $h(n, n')n'$ and $h(n, 1)n'$. Receiving a hash value α which is either $\mathcal{H}(1^\eta, \nu\nu')$ or $\mathcal{H}(1^\eta, \nu 1)$, D cannot just generate a nonce ν'' and call A on $\alpha\nu''$: the distribution of $\mathcal{H}(1^\eta, \nu\nu')\nu'$ is not equal to that of $\mathcal{H}(1^\eta, \nu\nu')\nu''$. The technical solution is to provide the adversary D with access to ν'; this is enough to prove that oracle indistinguishability can then be broken. What is still needed is that the inside of the hash is still "random enough" even if part of it is revealed; in this particular case this means that revealing ν', the inside of that hash is still hidden to D.

We now first prove, in general, that it is safe to reveal some of the randomness inside the hash. More accurately, if you pre-specify some, but not all, of the sequences of coins to be chosen when interpreting a message m, then no single bitstring x is exceptionally likely to occur as the interpretation of m. After this lemma, we can prove the soundness result.

Lemma 4.5.5. Let m be a message, $U \subsetneq \mathsf{R}(m)$. Let p be a positive polynomial. Then, for large enough η

$$\forall \tau \in \mathsf{Coins}_\eta(U). \forall x \in \mathbf{Str} : \mathbb{P}[\alpha \leftarrow [\![m]\!]_\eta^\tau; \alpha = x] < \frac{1}{p(\eta)}.$$

Proof. The proof follows by induction on the structure of m.

- Consider the case $m = \langle m_1, m_2 \rangle$. We get by induction hypothesis that the statement holds either for m_1 or m_2, which suffices for the proof given that concatenating a bitstring might just lower the probability of hitting a particular element.
- The cases $m = \{\!|m_1|\!\}_k^r$ and $m = \square^r$ are trivial due to well-spreadness (Definition 4.3.6).
- The cases $\boxtimes^r$ and $m = h^r(m_1)$ follow from collision resistance (Definition 4.3.9).
- The case $m = c$ does not occur since U must be a proper subset of $\mathsf{R}(c) = \emptyset$.

- If m is a nonce n, then $U = \emptyset$ since it must be a proper subset of $\mathsf{R}(n) = \{n\}$. Then $\mathbb{P}[\alpha \leftarrow [\![n]\!]^{\tau}_{\eta}; \alpha = x] = \frac{1}{2^{\eta}} < \frac{1}{p(\eta)}$.

- If m is a key k, then again $U = \emptyset$. Suppose that for infinitely many η there is a $\tau \in \mathsf{Coins}_{\eta}(U)$ and an $x \in \mathbf{Str}$ for which

$$\mathbb{P}[\alpha \leftarrow [\![k]\!]^{\tau}_{\eta}; \alpha = x] = \mathbb{P}[\alpha \leftarrow \mathcal{K}(1^{\eta}); \alpha = x] \geq \frac{1}{p(\eta)}. \tag{4.0}$$

  Now we build an adversary $A^{\mathcal{F}(-),\mathcal{G}(-)} : \mathbf{Param} \to \{0,1\}$ that breaks type-0 security.

algorithm $A^{\mathcal{F}(-),\mathcal{G}(-)}(1^{\eta})$:
$\nu \leftarrow \mathcal{N}(1^{\eta})$
$\epsilon \leftarrow \mathcal{F}(\nu)$
$\kappa \leftarrow \mathcal{K}(1^{\eta})$
if $\mathcal{D}(\kappa, \epsilon) = \nu$ **return** 1
else return 0

This adversary generates a random nonce ν and gives it to the oracle $\mathcal{F}$ to encrypt. The adversary tries to guess if the oracle was instantiated with $\mathcal{E}(k,-)$ or with $\mathcal{E}(k,0)$ by simply randomly generating a key itself and trying to decrypt. We will show that the probability that the oracle and the adversary choose the same key is non-negligible and hence the probability that this adversary guesses correctly is also non-negligible. Omitting $\mathcal{G}$ as it is not used by A, we get

$$
\begin{aligned}
&\mathsf{Adv}_A(\eta)\\
&\quad= \mathbb{P}[\kappa \leftarrow \mathcal{K}(1^\eta); \nu \leftarrow \mathcal{N}(1^\eta); \epsilon \leftarrow \mathcal{E}(\kappa,\nu); \kappa' \leftarrow \mathcal{K}(1^\eta); \mathcal{D}(\kappa',\epsilon) = \nu]\\
&\qquad - \mathbb{P}[\kappa \leftarrow \mathcal{K}(1^\eta); \nu \leftarrow \mathcal{N}(1^\eta); \epsilon \leftarrow \mathcal{E}(\kappa,0); \kappa' \leftarrow \mathcal{K}(1^\eta); \mathcal{D}(\kappa',\epsilon) = \nu]\\
&\quad\geq \mathbb{P}[\kappa,\kappa' \leftarrow \mathcal{K}(1^\eta); \kappa = \kappa']\\
&\qquad - \sum_{y\in\{0,1\}^\eta} \mathbb{P}[\nu \leftarrow \mathcal{N}(1^\eta); y = \nu] \cdot \mathbb{P}[\kappa,\kappa' \leftarrow \mathcal{K}(1^\eta); \mathcal{D}(\kappa',\mathcal{E}(\kappa,0)) = y]\\
&\qquad \text{(because it is always possible to decrypt with the proper key)}\\
&\quad\geq \frac{1}{p(\eta)^2} - 2^{-\eta} \sum_{y\in\{0,1\}^\eta} \mathbb{P}[\kappa,\kappa' \leftarrow \mathcal{K}(1^\eta); \epsilon \leftarrow \mathcal{E}(\kappa,0); \mathcal{D}(\kappa',\epsilon) = y]\\
&\qquad \text{(bounding the first term by the probability of getting } x \text{ two times)}\\
&\quad\geq \frac{1}{p(\eta)^2} - 2^{-\eta} \sum_{\kappa_0,\kappa_0',\epsilon_0} \Bigg(\mathbb{P}[\kappa,\kappa' \leftarrow \mathcal{K}(1^\eta); \kappa = \kappa_0; \kappa' = \kappa_0'; \mathcal{E}(\kappa,0) = \epsilon_0]\\
&\qquad\qquad \cdot \sum_{y\in\{0,1\}^\eta} \mathbb{P}[\mathcal{D}(\kappa_0',\epsilon_0) = y]\Bigg)\\
&\quad\geq \frac{1}{p(\eta)^2} - 2^{-\eta} \geq \frac{1}{p(\eta)^3} \qquad \text{(for large enough } \eta).
\end{aligned}
$$

□

Theorem 4.5.6. Let m be a message with a sub-message of the form $h^r(\tilde{m})$. Assume that $\tilde{m} \notin \overline{m}$. Take $m' := m[h^r(\tilde{m}) := \boxtimes^s]$, where $s = \mathcal{R}(h^r(\tilde{m}))$. Then $[\![m]\!] \equiv [\![m']\!]$.

Proof. Assume that $[\![m]\!] \not\equiv [\![m']\!]$, say $A\colon \mathbf{Param} \times \mathbf{Str} \to \{0,1\}$ is a probabilistic polynomial-time adversary and p a positive polynomial such that

$$\frac{1}{p(\eta)} \leq \mathbb{P}[\mu \leftarrow [\![m]\!]_\eta; A(1^\eta,\mu) = 1] - \mathbb{P}[\mu \leftarrow [\![m']\!]_\eta; A(1^\eta,\mu) = 1] \qquad (4.0)$$

for infinitely many $\eta \in \mathbb{N}$. We will use this to build a distinguisher as in Definition 4.3.8 that breaks oracle indistinguishability of $\langle \mathcal{H}, \mathcal{V} \rangle$.

Let $\eta \in \mathbb{N}$, abbreviate $\mathsf{R}(m, \tilde{m}) \cap \mathsf{R}(\tilde{m})$ to U and let $\tau \in \mathsf{Coins}_\eta(U)$. Note that τ chooses coin flips for the shared randomness between the inside and the outside of the hash. Then define a probabilistic polynomial-time algorithm $D_\eta^\tau \colon \{0,1\}^* \to \{0,1\}$ as follows.

algorithm $D_\eta^\tau(\alpha)$:
  $\mu \leftarrow [\![m]\!]_\eta^{\{h^r(\tilde{m}) \mapsto \alpha\}, \tau}$
  $\beta \leftarrow A(\eta, \mu)$
  return β

This algorithm tries to guess if a given bitstring α was drawn from $[\![h^r(\tilde{m})]\!]_\eta^\tau$ or from $[\![\boxtimes^s]\!]_\eta^\tau = [\![h^s(n_{\boxtimes}^s)]\!]_\eta^\tau$. It does so by computing an interpretation for m as follows. The sub-message $h^r(\tilde{m})$ is interpreted as α; the randomness that is shared between the inside of the hash ($\tilde{m}$) and the rest of the message is resolved using hard-coded sequences of coin flips τ. It then uses the adversary A to guess if the resulting interpretation was drawn from $[\![m]\!]_\eta$ (in which case it guesses that α was drawn from $[\![h^r(\tilde{m})]\!]_\eta$) or from $[\![m']\!]_\eta$ (in which case it guesses that α was drawn from $[\![\boxtimes^s]\!]_\eta$).

Note that for every η and τ we have a different algorithm D_η^τ that has hardcoded coin flips for the shared randomness. However we do not have a single algorithm taking η and α as arguments since such an algorithm would need an infinite sequence of hardcoded coin flips.

Now consider one of the infinitely many values of η for which (4.5) holds. Using D_η^τ we can rephrase (4.5) as follows:

$$\begin{aligned}
\frac{1}{p(\eta)} &\leq \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(U), \alpha \leftarrow [\![h^r(\tilde{m})]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] - \\
&\qquad\qquad \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(U), \alpha \leftarrow [\![\boxtimes^s]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] \\
&= \sum_{\tau \in \mathsf{Coins}_\eta(U)} \Big(\mathbb{P}[\alpha \leftarrow [\![h^r(\tilde{m})]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] - \\
&\qquad\qquad \mathbb{P}[\alpha \leftarrow [\![\boxtimes^s]\!]_\eta^\tau; D_\eta^\tau(\alpha) = 1] \Big) \cdot \mathbb{P}[T \leftarrow \mathsf{Coins}_\eta(U); T = \tau] \\
&= \sum_{\tau \in \mathsf{Coins}_\eta(U)} \Big(\mathbb{P}[\alpha \leftarrow [\![\tilde{m}]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] - \\
&\qquad\qquad \mathbb{P}[\alpha \leftarrow [\![n_{\boxtimes}^s]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] \Big) \cdot \mathbb{P}[T \leftarrow \mathsf{Coins}_\eta(U); T = \tau] \\
&= \frac{1}{|\mathsf{Coins}_\eta(U)|} \sum_{\tau \in \mathsf{Coins}_\eta(U)} \Big(\mathbb{P}[\alpha \leftarrow [\![\tilde{m}]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] - \\
&\qquad\qquad \mathbb{P}[\alpha \leftarrow [\![n_{\boxtimes}^s]\!]_\eta^\tau; D_\eta^\tau(\mathcal{H}(1^\eta, \alpha)) = 1] \Big).
\end{aligned}$$

Note that τ selects the randomness that is shared between the inside of the hash and the outside of the hash; when α is drawn from $[\![\tilde{m}]\!]_\eta^\tau$ the randomness that appears only inside the hash is chosen (and the assumption on $\tilde{m}$ means that there is really something to choose); $\mathcal{H}$ chooses the randomness for taking the hash; and D_η^τ itself resolves the randomness that appears only outside the hash.

This means that there must be a particular value of τ, say $\bar{\tau}_\eta$, such that

$$\frac{1}{p(\eta)} \leq \mathbb{P}[\alpha \leftarrow [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}; D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] - \mathbb{P}[\alpha \leftarrow [\![n_{\boxtimes}^s]\!]_\eta^{\bar{\tau}_\eta}; D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1].$$

Gathering all $D_\eta^{\bar{\tau}_\eta}$ together for the various values of η, let D be the non-uniform adversary $\{D_\eta^{\bar{\tau}_\eta}\}_{\eta \in \mathbb{N}}$. Note that we have not actually defined $D_\eta^{\bar{\tau}_\eta}$ for all η, but only for those (infinitely many) for which (4.5) actually holds. What D does for the other values of η is irrelevant.

We will now show that D breaks the oracle indistinguishability of $\langle \mathcal{H}, \mathcal{V} \rangle$. For this, let $L = \{L_\eta\}_{\eta \in \mathbb{N}}$ be a polynomial size family of subsets of $\mathbf{Str}$. We have to show that for infinitely many values of η, there are $x, y \in \mathbf{Str} \setminus L_\eta$ such that D meaningfully distinguishes between $\mathcal{H}(1^\eta, x)$ and $\mathcal{H}(1^\eta, y)$.

Once again, take one of the infinitely many values of η for which (4.5) holds. Continuing from (4.5), we get

$$\begin{aligned}
\frac{1}{p(\eta)} \leq & \sum_{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}} \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \\
& \qquad - \sum_{\beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}} \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \beta)) = 1] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \\
= & \sum_{\substack{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \\ \beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}}} \Big[\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \\
& \qquad - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \beta)) = 1] \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \Big] \\
& \text{(since } \sum_{\beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}} \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] = 1 \text{ and } \sum_{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}} \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] = 1) \\
= & \sum_{\substack{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \\ \beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} \cap L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \beta)) = 1] \big) \\
& \qquad \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \Big] \\
& \quad + \sum_{\substack{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \cap L_\eta \\ \beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \beta)) = 1] \big) \\
& \qquad \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \Big] \\
& \quad + \sum_{\substack{\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta \\ \beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \alpha)) = 1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta, \beta)) = 1] \big) \\
& \qquad \cdot \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} = \alpha] \cdot \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} = \beta] \Big] \\
& \text{(splitting cases on } \in L_\eta \text{ and } \notin L_\eta)
\end{aligned}$$

$$\leq \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\cap L_\eta}} \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]\cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]$$

$$+ \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\cap L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]\cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]$$

$$+ \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]-\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]\big) \cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]\Big]$$

(since $\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]\geq 0$)

$$\leq \sum_{\beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\cap L_\eta} \mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta] + \sum_{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\cap L_\eta} \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]$$

$$+ \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]-\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]\big) \cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]\Big]$$

(since $\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]\leq 1$, $\sum_{\alpha\in[\![m]\!]_\eta^{\bar{\tau}_\eta}}\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]=1$ and $\sum_{\beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]\leq 1$)

$$\leq \frac{1}{2p(\eta)} + \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \Big[\big(\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]-\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]\big) \cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta]\Big].$$

(by Lemma 4.5.5 (2x) using the polynomial $4p(\eta)|L_\eta|$, provided that η is large)

Now suppose that for all $\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta$ and all $\beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta$ we have

$$\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1]-\mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1]<\frac{1}{2p(\eta)}.$$

Then, continuing from (4.5), we get a contradiction:

$$\begin{aligned}\frac{1}{p(\eta)} &< \frac{1}{2p(\eta)} + \sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \frac{1}{2p(\eta)}\cdot\mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta] \\ &= \frac{1}{2p(\eta)} + \frac{1}{2p(\eta)}\sum_{\substack{\alpha\in[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta \\ \beta\in[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}\setminus L_\eta}} \mathbb{P}[[\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta}=\alpha]\cdot\mathbb{P}[[\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta}=\beta] \\ &\leq \frac{1}{2p(\eta)} + \frac{1}{2p(\eta)}\,.\end{aligned}$$

Therefore, there must be an $\alpha \in [\![\tilde{m}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ and a $\beta \in [\![n^s_{\boxtimes}]\!]_\eta^{\bar{\tau}_\eta} \setminus L_\eta$ such that

$$\frac{1}{2p(\eta)} \leq \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\alpha))=1] - \mathbb{P}[D_\eta^{\bar{\tau}_\eta}(\mathcal{H}(1^\eta,\beta))=1].$$

Hence D breaks oracle indistinguishability, contradicting the assumption on $\langle\mathcal{H},\mathcal{V}\rangle$. □

Theorem 4.5.7 (Soundness)**.** Let m and m' be acyclic messages. Then $m \cong m' \implies [\![m]\!] \equiv [\![m']\!]$.

Proof. The assumption that $m \cong m'$ means that $\mathsf{pattern}(m) \approx \mathsf{pattern}(m')$. Therefore we get $[\![\mathsf{pattern}(m)]\!] = [\![\mathsf{pattern}(m')]\!]$. By lemma 4.5.3 we get $[\![\mathsf{encpat}\circ\mathsf{hashpat}(m)]\!] = [\![\mathsf{encpat}\circ\mathsf{hashpat}(m')]\!]$. Now, by applying Theorem 4.5.4 two times, we obtain $[\![\mathsf{hashpat}(m)]\!] \equiv [\![\mathsf{hashpat}(m')]\!]$. Finally, we start with m and repeatedly apply Theorem 4.5.6, replacing one hash at a time by $\boxtimes$, and arrive at $\mathsf{hashpat}(m)$. This shows that $[\![m]\!] \equiv [\![\mathsf{hashpat}(m)]\!]$ and similarly $[\![m']\!] \equiv [\![\mathsf{hashpat}(m')]\!]$. Therefore $[\![m]\!] \equiv [\![m']\!]$. □

4.6 Completeness

Although soundness results allow us to port proofs of *secrecy* properties from the symbolic world to the computational world, it does not permit to port, for instance, *authenticity* and *integrity* results. For example, consider

a protocol in which an agent A chooses a nonce n and commits to this nonce by sending $h(n)$ to another agent B. Later in the protocol, A will reveal the nonce n by sending n itself to B. Security in this setting means that A cannot change her choice after sending $h(n)$. In the symbolic world, this is guaranteed by the fact that the message $h(n)n$ (the concatenation of the relevant messages in the protocol run) is observationally distinct from $h(n)n'$, with $n' \neq n$. We would like to be able to conclude from this symbolic property that $[\![h(n)n]\!]$ is computationally distinct from $[\![h(n)n']\!]$, since that is needed to guarantee the security in the computational world.

What is needed here is completeness: computational equivalence of $[\![m]\!]$ and $[\![m']\!]$ should imply observational equivalence of m and m'. For the original Abadi-Rogaway logic, completeness under appropriate conditions on the encryption scheme was proven by Micciancio and Warinschi [MW04a].

This section now shows that the interpretation proposed in the Section 4.4 is complete. Throughout this section we assume that the encryption scheme $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ is type-0 secure, well-spread, and confusion free, and that the probabilistic hash scheme $\langle \mathcal{H}, \mathcal{V} \rangle$ is collision resistant and oracle indistinguishable.

Throughout the completeness proof we follow the steps of Micciancio–Warinschi and their notation when possible. We recall here some of their results as they are used in our proof.

In the original Abadi-Rogaway logic, the useful information for an adversary is determined by the set of keys she can learn. We define the function $\mathsf{arecover}$ and its computational counterpart $\mathsf{brecover}$ as in [MW04a]. There they are called *recoverable* and *getkeys*. These functions extract the set of keys observable by an adversary from a symbolic message and a bitstring respectively.

Definition 4.6.1. The function $\mathsf{arecover}\colon \mathbf{Msg} \to \mathcal{P}(\mathbf{Key})$ is defined by:

$$\mathsf{arecover}(m) = \mathsf{arecover}(m, |m|)$$

where

$$\mathsf{arecover}(m, 0) = \emptyset$$
$$\mathsf{arecover}(m, d+1) = \mathsf{F}_{\mathsf{kr}}(m, \mathsf{arecover}(m, d))$$

and

$$\begin{aligned}
\mathsf{F}_{\mathsf{kr}}(\langle m_1, m_2\rangle, U) &= \mathsf{F}_{\mathsf{kr}}(m_1, U) \cup \mathsf{F}_{\mathsf{kr}}(m_2, U) \\
\mathsf{F}_{\mathsf{kr}}(k, U) &= \{k\} \cup U \\
\mathsf{F}_{\mathsf{kr}}(\{\!|m|\!\}_k^r, U) &= \mathsf{F}_{\mathsf{kr}}(m, U), && \text{if } k \in U; \\
\mathsf{F}_{\mathsf{kr}}(m, U) &= U, && \text{in any other case.}
\end{aligned}$$

The function $\mathsf{brecover}\colon \mathbf{Str} \to \mathcal{P}(\mathbf{Keys})$ is defined by

algorithm $\mathsf{brecover}(\mu)$:
Gets all the keys in the bitstring μ with high probability.
  $U' := \emptyset$
  do
    $U := U'$
    $U' := \mathsf{C}_{\mathsf{kr}}(\mu, U)$
  until $U = U'$
  return U

where

$$\begin{aligned}
\mathsf{C}_{\mathsf{kr}}(\kappa, U) &= \{\kappa\} \cup U \\
\mathsf{C}_{\mathsf{kr}}(\mu_1\mu_2, U) &= \mathsf{C}_{\mathsf{kr}}(\mu_1, U) \cup \mathsf{C}_{\mathsf{kr}}(\mu_2, U) \\
\mathsf{C}_{\mathsf{kr}}(\epsilon, U) &= \mathsf{C}_{\mathsf{kr}}(\mu, U), \text{if } \exists!\kappa \in U \text{ s.t. } \mathcal{D}(\epsilon, \kappa) = \mu \neq \bot; \\
\mathsf{C}_{\mathsf{kr}}(\mu, U) &= U, \quad \text{otherwise.}
\end{aligned}$$

Note that brecover can be computed in polynomial time: we can assume without loss of generality that the size of the output of the decryption algorithm is smaller than the input ciphertext (since the encryption scheme is randomized). Then C_{kr} recurses a linear number of times and every iteration of the until loop of brecover adds at least one key to U' and therefore the number of iterations is bounded by the maximum number of keys in μ.

The following lemma also from [MW04a] shows the relation between these two functions.

Lemma 4.6.2. Let $\Pi = \langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ be a confusion free encryption scheme and let $m \in \mathbf{Msg}$. Then

$$\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{brecover}([\![m]\!]_\eta^\tau) \neq [\![\mathsf{arecover}(m)]\!]_\eta^\tau]$$

is a negligible function of η.

Proof. We refer the reader to the original paper for a complete proof of this lemma. The hashes that appear in our logic have no influence at all. □

In our extended logic, due to the hashes, it is not true any more that the only useful information for an adversary is the set of keys. Any message an adversary can learn might be the pre-image of a certain hash value. Therefore, we need to be able to compute the closure (up to a certain size) of a given message or bitstring. For this reason the closure operators defined below are closed (up to a certain size) under pairing but not under encryption or hashing.

Definition 4.6.3. The function $\mathsf{aclosure}\colon \mathbf{Msg} \times \mathbb{N} \to \mathcal{P}(\mathbf{Msg})$ computes the messages in the symbolic closure of a message, up to a certain size d:

$$\mathsf{aclosure}(m, d) = \mathsf{aclosure}(m, d, \mathsf{arecover}(m))$$

where

$$\mathsf{aclosure}(m, d, U) = \mathsf{asynth}(\mathsf{aanalz}(m, U), d).$$

Here the function $\mathsf{aanalz}\colon \mathbf{Msg} \times \mathcal{P}(\mathbf{Msg}) \to \mathcal{P}(\mathbf{Msg})$, defined below, splits a message in all its meaningful subterms, using the keys in U, when possible, for decrypting.

$$\begin{aligned} \mathsf{aanalz}(\langle m_1, m_2 \rangle, U) &= \mathsf{aanalz}(m_1, U) \cup \mathsf{aanalz}(m_2, U) \\ \mathsf{aanalz}(\{\!|m|\!\}_k^r, U) &= \{\{\!|m|\!\}_k^r\} \cup \mathsf{aanalz}(m, U), && \text{if } k \in U; \\ \mathsf{aanalz}(m, U) &= \{m\} \end{aligned}$$

The function $\mathsf{asynth}\colon \mathcal{P}(\mathbf{Msg}) \times \mathbb{N} \to \mathcal{P}(\mathbf{Msg})$ generates all possible vectors of messages in U of size up to d:

algorithm $\mathsf{asynth}(U, d)$:
  $U_1 = U$
  for $i = 2$ **to** d
    $U_i := \emptyset$
    for each $m \in U$
      for each $v \in U_{i-1}$
        $U_i := U_i \cup \{\langle m, v\rangle\}$
  return U_d

algorithm $\mathsf{bsynth}(U, d)$:
  $U_1 = U$
  for $i = 2$ **to** d
    $U_i := \emptyset$
    for each $\mu \in U$
      for each $\omega \in U_{i-1}$
        $U_i := U_i \cup \{\mu\omega\}$
  return U_d

Next, the functions $\mathsf{bclosure}\colon \mathbf{Str}\times\mathbb{N} \to \mathcal{P}(\mathbf{Str})$, $\mathsf{banalz}\colon \mathbf{Str}\times\mathcal{P}(\mathbf{Str}) \to \mathcal{P}(\mathbf{Str})$ and $\mathsf{bsynth}\colon \mathcal{P}(\mathbf{Str}) \times \mathbb{N} \to \mathcal{P}(\mathbf{Str})$ are the computational counterparts of $\mathsf{aclosure}$, aanalz and asynth respectively:

$$\mathsf{bclosure}(\mu, d) = \mathsf{bclosure}(\mu, d, \mathsf{brecover}(\mu))$$

where

$$\mathsf{bclosure}(\mu, d, U) = \mathsf{bsynth}(\mathsf{banalz}(\mu, U), d)$$

and

$$\begin{aligned}
\mathsf{banalz}(\mu_1\mu_2, U) &= \mathsf{banalz}(\mu_1, U) \cup \mathsf{banalz}(\mu_2, U)\\
\mathsf{banalz}(\epsilon, U) &= \{\epsilon\} \cup \mathsf{banalz}(\mu, U), \quad \text{if } \exists!\kappa \in U s.t. \mathcal{D}(\epsilon, \kappa) = \mu \neq \bot;\\
\mathsf{banalz}(\mu, U) &= \{\mu\}
\end{aligned}$$

Note that for a fixed $d \in \mathbb{N}$, $\mathsf{bclosure}(\mu, d)$ can be computed in a time that is polynomial in $|\mu|$.
Now we show that the proposed functions behave similarly with high probability.

Lemma 4.6.4. Let $m \in \mathbf{Msg}$, $d \in \mathbb{N}$ and $T \subseteq \mathbf{Key}$. Then the probability

$$\mathbb{P}\left[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{bclosure}([\![m]\!]_\eta^\tau, d, [\![T]\!]_\eta^\tau) \neq [\![\mathsf{aclosure}(m, d, T)]\!]_\eta^\tau\right]$$

is a negligible function of η.

Proof. We prove by induction on the structure of m that the probability

$$\mathbb{P}\left[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) \neq \llbracket \mathsf{aanalz}(m, T) \rrbracket_\eta^\tau\right]$$

is a negligible function of η. The original statement follows from this, using that asynth and *bsynth* have similar behavior.

The only non-trivial case is that of $m = \{\!|m_1|\!\}_k^r$.

• If $k \in T$, then $\llbracket k \rrbracket_\eta^\tau \in \llbracket T \rrbracket_\eta^\tau$. Next

$$\begin{aligned}
&\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) = \llbracket \mathsf{aanalz}(m, T) \rrbracket_\eta^\tau] \\
&\quad = \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) = \llbracket \{m\} \cup \mathsf{aanalz}(m_1, T) \rrbracket_\eta^\tau] \\
&\quad \geq \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \llbracket m \rrbracket_\eta^\tau \cup \mathsf{banalz}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) = \llbracket \{m\} \cup \mathsf{aanalz}(m_1, T) \rrbracket_\eta^\tau \\
&\qquad\qquad \wedge \forall \kappa \in \llbracket T \setminus k \rrbracket_\eta^\tau : \mathcal{D}(\llbracket m \rrbracket_\eta^\tau, \kappa) = \bot] \\
&\quad \geq \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) = \llbracket \mathsf{aanalz}(m_1, T) \rrbracket_\eta^\tau \\
&\qquad\qquad \wedge \forall \kappa \in \llbracket T \setminus k \rrbracket_\eta^\tau : \mathcal{D}(\llbracket m \rrbracket_\eta^\tau, \kappa) = \bot] \\
&\quad \geq 1 - (\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) \neq \llbracket \mathsf{aanalz}(m_1, T) \rrbracket_\eta^\tau \\
&\qquad\qquad \vee \exists \kappa \in \llbracket T \setminus k \rrbracket_\eta^\tau : \mathcal{D}(\llbracket m \rrbracket_\eta^\tau, \kappa) \neq \bot]) \\
&\quad \geq 1 - (\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket T \rrbracket_\eta^\tau) \neq \llbracket \mathsf{aanalz}(m_1, T) \rrbracket_\eta^\tau] \\
&\qquad\qquad + \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \exists \kappa \in \llbracket T \setminus k \rrbracket_\eta^\tau : \mathcal{D}(\llbracket m \rrbracket_\eta^\tau, \kappa) \neq \bot]) \\
&\quad \geq 1 - (\varepsilon_1(\eta) + \sum_{\kappa \in \llbracket T \setminus k \rrbracket_\eta^\tau} \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathcal{D}(\llbracket m \rrbracket_\eta^\tau, \kappa) \neq \bot]) \\
&\quad \geq 1 - (\varepsilon_1(\eta) + \varepsilon_2(\eta) \cdot (|T| - 1)),
\end{aligned}$$

where $\varepsilon_1, \varepsilon_2$ are the negligible functions from the induction hypothesis and confusion freeness respectively.

• If $k \notin T$, then $[\![k]\!]^\tau_\eta \notin [\![T]\!]^\tau_\eta$. Next

$$\begin{aligned}
&\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}([\![m]\!]^\tau_\eta, [\![T]\!]^\tau_\eta) = [\![\mathsf{aanalz}(m,T)]\!]^\tau_\eta] \\
&\quad = \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}([\![m]\!]^\tau_\eta, [\![T]\!]^\tau_\eta) = [\![\{m\}]\!]^\tau_\eta] \\
&\quad \geq \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{banalz}([\![m]\!]^\tau_\eta, [\![T]\!]^\tau_\eta) = [\![\{m\}]\!]^\tau_\eta \\
&\qquad\qquad \wedge \forall \kappa \in [\![T]\!]^\tau_\eta : \mathcal{D}([\![m]\!]^\tau_\eta, \kappa) = \bot] \\
&\quad = \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); [\![m]\!]^\tau_\eta = [\![\{m\}]\!]^\tau_\eta \wedge \forall \kappa \in [\![T]\!]^\tau_\eta : \mathcal{D}([\![m]\!]^\tau_\eta, \kappa) = \bot] \\
&\quad = 1 - \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \exists \kappa \in [\![T]\!]^\tau_\eta : \mathcal{D}([\![m]\!]^\tau_\eta, \kappa) = \bot] \\
&\quad \geq 1 - \sum_{\kappa \in [\![T]\!]^\tau_\eta} \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathcal{D}([\![m]\!]^\tau_\eta, \kappa) = \bot] \\
&\quad \geq 1 - \varepsilon(\eta) \cdot |T| \,,
\end{aligned}$$

where ε is a negligible function due to confusion freeness. □

The following is an extended version of the function psp from [MW04a], which is the computational counterpart of pattern. This function takes a bitstring as an argument and tries to recover the pattern associated with it. This means that given as input a sample from $[\![m]\!]$, the function outputs (a renaming of) $\mathsf{pattern}(m)$ with high probability. As in [MW04a] we let f be an arbitrary (but fixed) injective function that associates an identifier (i.e., an element of $\mathbf{Nonce} \cup \mathbf{Key} \cup \mathbf{Const}$) to each bitstring of primitive type (i.e., ν, κ, ς).

$$\begin{aligned}
\mathsf{psp}(\mu_1\mu_2, U) &= \langle \mathsf{psp}(\mu_1, U), \mathsf{psp}(\mu_2, U) \rangle \\
\mathsf{psp}(\epsilon, U) &= \begin{cases} \{\!|\mathsf{psp}(\mathcal{D}(\epsilon,\kappa), U)|\!\}^{\mathcal{R}(\epsilon)}_{f(\kappa)}, & \text{if } \exists! \kappa \in U \text{ s.t. } \mathcal{D}(\epsilon,\kappa) \neq \bot; \\ \square^{\mathcal{R}(\epsilon)}, & \text{otherwise.} \end{cases} \\
\mathsf{psp}(\psi, U) &= \begin{cases} h^{\mathcal{R}(\psi)}(\mathsf{psp}(\mu, U)), & \text{if } \exists! \mu \in U \text{ s.t. } \mathcal{V}(\mu,\psi) = 1; \\ \boxtimes^{\mathcal{R}(\psi)}, & \text{otherwise.} \end{cases} \\
\mathsf{psp}(\mu, U) &= f(\mu) \qquad \text{in any other case.}
\end{aligned}$$

Theorem 4.6.5. Let $m \in \mathbf{Msg}$ and let U be a finite subset of $\mathbf{Msg}$. Then the probability

$$\mathbb{P}\left[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \not\approx \mathsf{pattern}(m, U)\right]$$

is a negligible function of η.

Proof. The proof follows by induction on the structure of m. We only show here the case $m = h^r(m_1)$. For the remaining cases, the proof follows similarly to the one in the original Micciancio-Warinschi [MW04a] paper and therefore we refer the reader to it.

• If $m_1 \in U$ then

$$\begin{aligned}
&\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \approx \mathsf{pattern}(m, U)] \\
&\quad \geq \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \approx \mathsf{pattern}(m, U) \\
&\qquad\qquad \wedge \forall \mu \in \llbracket U \setminus \{m_1\} \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 0] \\
&\quad = \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \approx \mathsf{pattern}(m_1, U) \\
&\qquad\qquad \wedge \forall \mu \in \llbracket U \setminus \{m_1\} \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 0] \\
&\quad = 1 - \mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \not\approx \mathsf{pattern}(m_1, U) \\
&\qquad\qquad \vee \exists \mu \in \llbracket U \setminus \{m_1\} \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 1] \\
&\quad \geq 1 - \mathbb{P}\left[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m_1 \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \not\approx \mathsf{pattern}(m_1, U)\right] \\
&\qquad\qquad + \mathbb{P}\left[\tau \leftarrow \mathsf{Coins}_\eta(m); \exists \mu \in \llbracket U \setminus \{m_1\} \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 1\right] \\
&\quad \geq 1 - \varepsilon_1(\eta) - \varepsilon_2(\eta)\,,
\end{aligned}$$

where ε_1 is the negligible function from the induction hypothesis and ε_2 is a negligible function from collision resistance, using that an adversary can compute $\llbracket U \setminus \{m_1\} \rrbracket_\eta^\tau$.

• If $m_1 \notin U$ then

$$\begin{aligned}
&\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \approx \mathsf{pattern}(m, U)] \\
&\qquad = \mathbb{P}[\mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \approx \boxtimes^r] = \mathbb{P}[\forall \mu \in \llbracket U \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 0]
\end{aligned}$$

therefore

$$\begin{aligned}&\mathbb{P}[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket U \rrbracket_\eta^\tau) \not\approx \mathsf{pattern}(m, U)]\\ &\quad = \mathbb{P}[\exists \mu \in \llbracket U \rrbracket_\eta^\tau : \mathcal{V}(\mu, \llbracket m \rrbracket_\eta^\tau) = 1] \leq \varepsilon(\eta)\,,\end{aligned}$$

where ε is a negligible function due to collision resistance. □

Lemma 4.6.6. Let $m \in \mathbf{Msg}$ and $d \in \mathbb{N}$. Then the probability

$$\mathbb{P}\left[\mu \leftarrow \llbracket m \rrbracket; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu, d)) \not\approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\right]$$

is a negligible function of η.

Proof.

$$\begin{aligned}&\mathbb{P}\big[\mu \leftarrow \llbracket m \rrbracket; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu, d)) \approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\big]\\
&\quad = \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d)) \approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\big]\\
&\quad \geq \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d)) \approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\\
&\qquad\qquad \wedge \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d) = \llbracket \mathsf{aclosure}(m, d) \rrbracket_\eta^\tau\big]\\
&\quad = \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket \mathsf{aclosure}(m, d) \rrbracket_\eta^\tau) \approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\\
&\qquad\qquad \wedge \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d) = \llbracket \mathsf{aclosure}(m, d) \rrbracket_\eta^\tau\big]\\
&\quad \geq 1 - \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{psp}(\llbracket m \rrbracket_\eta^\tau, \llbracket \mathsf{aclosure}(m, d) \rrbracket_\eta^\tau) \not\approx \mathsf{pattern}(m, \mathsf{aclosure}(m, d))\big]\\
&\qquad\qquad - \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d) \neq \llbracket \mathsf{aclosure}(m, d) \rrbracket_\eta^\tau\big]\\
&\quad \geq 1 - \varepsilon_1(\eta) - \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d, \mathsf{brecover}(\llbracket m \rrbracket_\eta^\tau))\\
&\qquad\qquad \neq \llbracket \mathsf{aclosure}(m, d, \mathsf{arecover}(m)) \rrbracket_\eta^\tau\big]\end{aligned}$$

(where ε_1 is the negligible function due to Theorem 4.6.5)

$$\begin{aligned}&\geq 1 - \varepsilon_1(\eta) - \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m);\\
&\qquad \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d, \mathsf{brecover}(\llbracket m \rrbracket_\eta^\tau)) \neq \llbracket \mathsf{aclosure}(m, d, \mathsf{arecover}(m)) \rrbracket_\eta^\tau\\
&\qquad\qquad \vee \llbracket \mathsf{arecover}(m) \rrbracket_\eta^\tau \neq \mathsf{brecover}(\llbracket m \rrbracket_\eta^\tau)\big]\\
&\geq 1 - \varepsilon_1(\eta) - \mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m);\\
&\qquad \mathsf{bclosure}(\llbracket m \rrbracket_\eta^\tau, d, \llbracket \mathsf{arecover}(m) \rrbracket_\eta^\tau) \neq \llbracket \mathsf{aclosure}(m, d, \mathsf{arecover}(m)) \rrbracket_\eta^\tau\\
&\qquad\qquad \vee \llbracket \mathsf{arecover}(m) \rrbracket_\eta^\tau \neq \mathsf{brecover}(\llbracket m \rrbracket_\eta^\tau)\big]\end{aligned}$$

$$\begin{aligned}
&\geq 1-\varepsilon_1(\eta)-\mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m);\\
&\qquad \mathsf{bclosure}(\llbracket m\rrbracket^\tau_\eta, d, \llbracket \mathsf{arecover}(m)\rrbracket^\tau_\eta) \neq \llbracket \mathsf{aclosure}(m,d,\mathsf{arecover}(m))\rrbracket^\tau_\eta\big]\\
&\qquad\qquad -\mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m); \llbracket \mathsf{arecover}(m)\rrbracket^\tau_\eta \neq \mathsf{brecover}(\llbracket m\rrbracket^\tau_\eta)\big]\\
&\geq 1-\varepsilon_1(\eta)-\varepsilon_2(\eta)-\mathbb{P}\big[\tau \leftarrow \mathsf{Coins}_\eta(m);\\
&\qquad \mathsf{bclosure}(\llbracket m\rrbracket^\tau_\eta, d, \llbracket \mathsf{arecover}(m)\rrbracket^\tau_\eta) \neq \llbracket \mathsf{aclosure}(m,d,\mathsf{arecover}(m))\rrbracket^\tau_\eta\big]
\end{aligned}$$

(where ε_2 is the negligible function due to Lemma 4.6.2)

$$\geq 1-\varepsilon_1(\eta)-\varepsilon_2(\eta)-\varepsilon_3(\eta)\,,$$

where ε_3 is the negligible function due to Lemma 4.6.4. □

Theorem 4.6.7 (Completeness)**.** Let m_1 and m_2 be acyclic messages. Then $\llbracket m_1\rrbracket \equiv \llbracket m_2\rrbracket \implies m_1 \cong m_2$.

Proof. Let us assume that $m_1 \ncong m_2$. Now we show that $\llbracket m_1\rrbracket \not\equiv \llbracket m_2\rrbracket$ by building a distinguisher D.

```
algorithm D(μ) :
  d := max(|m1|, |m2|)
  if psp(μ, bclosure(μ, d)) ≈ pattern(m1)
    return 1
  else
    return 0
```

Next we show that $\mathsf{Adv}_D(\eta) = |\mathbb{P}[\mu \leftarrow \llbracket m_1\rrbracket_\eta; D(\mu)=1] - \mathbb{P}[\mu \leftarrow \llbracket m_2\rrbracket_\eta; D(\mu)=1]|$ is not negligible. On the one hand

$$\begin{aligned}
&\mathbb{P}[\mu \leftarrow \llbracket m_1\rrbracket_\eta; D(\mu)=1]\\
&\qquad = \mathbb{P}[\mu \leftarrow \llbracket m_1\rrbracket_\eta; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu,d)) \approx \mathsf{pattern}(m_1)]\\
&\qquad = 1-\mathbb{P}[\mu \leftarrow \llbracket m_1\rrbracket_\eta; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu,d)) \not\approx \mathsf{pattern}(m_1)]\\
&\qquad \geq 1-\varepsilon_1(\eta)\,,
\end{aligned}$$

where ε_1 is the negligible function from Lemma 4.6.6. Note that $\mathsf{pattern}(m_1) = \mathsf{pattern}(m_1, \mathsf{aclosure}(m_1, |m_1|))$. On the other hand

$$\begin{aligned}\mathbb{P}[\mu \leftarrow [\![m_2]\!]_\eta; D(\mu) = 1] \\ &= \mathbb{P}[\mu \leftarrow [\![m_2]\!]_\eta; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu, d)) \approx \mathsf{pattern}(m_1)] \\ &\leq \mathbb{P}[\mu \leftarrow [\![m_2]\!]_\eta; \mathsf{psp}(\mu, \mathsf{bclosure}(\mu, d)) \not\approx \mathsf{pattern}(m_2)] \\ &\leq \varepsilon_2(\eta)\,,\end{aligned}$$

where ε_2 is the negligible function from Lemma 4.6.6. Therefore, $\mathsf{Adv}_D(\eta) = 1 - \varepsilon_1(\eta) - \varepsilon_2(\eta)$, which is not negligible. □

4.7 Active adversaries

We now briefly turn our attention to active adversaries. One could view the passive adversaries we have been considering up to now as being given the transcript of a protocol run and trying to deduce information from that. The soundness and completeness results say that the information an adversary could learn in the computational setting is the same as in the symbolic one. Active adversaries, however, can also try to inject messages in the network while the protocol is running. Hence, to obtain a soundness result, every meaningful message that an adversary could send in the computational world, should also be sendable in the symbolic world. Just as IND-CPA is not strong enough for encryption schemes for this to hold — one needs non-malleability [MW04b, Her05, MP05] — oracle hashing is not strong enough for hashes.

We now show an explicit example of an oracle hash scheme where an adversary is capable of creating more messages in the computational world than in the symbolic world.

Let $p = 2q + 1$ be a large (i.e., with $q > 2^\eta$) safe prime. Consider the oracle hash $\mathcal{H}(x) = \langle r^2, r^{2 \cdot f(x)} \mod p \rangle$ from Section 4.3.3. Assume that f is homomorphic for arguments up to length 2η, i.e., $f(x + y) = f(x)f(y) \mod q$ when $x+y < 2^\eta$. For instance, one could take $f(x) = g^x \mod q$ when $x < 2^\eta$ and $f(x) = h(x)$ otherwise, assuming that q is also a safe prime,

g is a generator of the quadratic residues modulo q, and h is a collision resistant one-way function. For simplicity, ignore the tagged representation (see Section 4.4) and assume that the representation of the concatenation of two nonces is just $\nu'\nu = 2^\eta \cdot \nu' + \nu$. Then

$$\begin{aligned}
\mathcal{H}(\nu'\nu) &= \mathcal{H}(2^\eta \cdot \nu' + \nu) \\
&= (r^2, r^{2 \cdot f(2^\eta \cdot \nu' + \nu)} \mod p) && \text{(for some } r\text{)} \\
&= (r^2, r^{2 \cdot f(\nu')^{2^\eta} f(\nu)} \mod p) && \text{(since } r^{2q} = 1 \mod p\text{)} \\
&= (r^2, (r^{2 \cdot f(\nu)})^{f(\nu')^{2^\eta}} \mod p).
\end{aligned}$$

Therefore, in the computational world with this particular oracle hash, an attacker receiving $\mathcal{H}(\nu) = (r^2, r^{2 \cdot f(\nu)})$ is capable of producing $\mathcal{H}(\nu'\nu)$ for a nonce ν' of her own choice. In the symbolic world, however, this is impossible since $h^r(n', n)$ is not in the closure of $\{h^r(n), n'\}$.

Next, we show a very simple one-way authentication protocol that, implemented with a malleable oracle hash function, results in a broken implementation. In this protocol, principal B authenticates to principal A, with whom he shares a symmetric key k_{AB}. The protocol is the following

1. $A \to B : n$
2. $B \to A : h(k_{AB}, n)$

Consider a homomorphic implementation $\mathcal{H}$ of h, as before. Now suppose that the attacker sees a protocol run: A sends to B a nonce ν, then B replies $\mathcal{H}(\kappa_{AB}, \nu)$. Later, the attacker is able to answer a new challenge ν' by sending $\mathcal{H}(\kappa_{AB}, \nu) \cdot \mathcal{H}(\nu' - \nu)$ to A. This results in a successful impersonation of B by the attacker.

The conclusion is that oracle hashing is not strong enough to give a perfect correspondence between the symbolic world and the computational world when the adversary is active. Just as for encryption schemes, what would be needed is a concept of *non-malleability* [DDN91] for hashes.

4.8 Conclusions

We have proposed an interpretation for formal hashes that is computationally sound and complete in the standard model. Under standard assumptions on hashes (pre-image and collision resistance), the symbolic world does not perfectly match the computational world. However, our results show that it is still possible to achieve this perfect match, for passive adversaries, using Canetti's oracle hashing. While considering active adversaries, we have shown that the security definition for oracle hashing is not strong enough. It would be interesting to extend the notion of non-malleability for hashes to achieve this perfect match also for active adversaries.

Chapter 5

Extending Computational Soundness Further: the case of Non-malleable Commitments

5.1 Introduction

Over the last few decades, two main stream approaches have been developed for the analysis of security protocols. On the one hand, the cryptographic approach considers an arbitrary computationally-bound adversary that interacts with honest participants and tries to break a security goal. This model is satisfactory as it deals with every efficient attacker. On the other hand, the logic or symbolic approach idealizes the security properties of the cryptographic primitives, which are axiomatized in a logic. Moreover, the capabilities of the adversary are also specified by a set of inference rules. This approach is appealing because there are automated techniques for the verification of some security properties.

This chapter further relates this two approaches considering commitment schemes in the presence of active adversaries. Recall from the intro-

duction and the previous chapter that this map relates messages that are observationally equivalent in the symbolic world to indistinguishable distributions over bitstrings. Such a map allows one to use formal methods, possibly even automated, to reason about security properties of protocols and have those reasonings be valid also in the computational setting.

Several extensions to the original Abadi-Rogaway logic [AR02] have been proposed in the literature. These extensions deal with public key encryption [MW04b, Her05], key cycles [ABHS05], partial information leakage [ABS05], active instead of passive adversaries [MW04b, JLM05], and more realistic security notions [AW05]. Other extensions add new primitives to the logic such as bilinear pairings [Maz07], modular exponentiation [BLMW07] and hash functions [CKKW06, GvR06b]. There are also frameworks dealing with generic equational theories [BCK05, ABW06, KM07]. So far there is no work in the literature, that we are aware of, that relates these two approaches for commitment schemes.

Commitment schemes are fundamental cryptographic primitives and are used in protocols like zero-knowledge proofs [GMW91], contract signing [EGL85], and can be used for bidding protocols. A commitment consists of two phases: the commitment phase where the principals commit to a message without revealing any information; and the opening phase where the principals reveal the message and it is possible to verify that this message corresponds to the value committed to during the commitment phase. After the commitment phase it should be infeasible to open the commitment to a different value than the one committed. This property is called binding. In the context of bidding protocols, non-malleability is also a desirable property. This means that an adversary cannot modify an intercepted commitment, say into a commitment to a slightly higher bid.

Our contribution. The first objective of this chapter is to find sufficient security assumptions to give a sound computational interpretation of commitments schemes in the Dolev-Yao model, under active adversaries. Pursuing that objective we propose a new indistinguishability-based security definition for commitment schemes in the presence of adaptive adversaries.

Then we give a novel generic construction for a non-malleable commitment scheme based on one-way trapdoor permutations. This construction is secure with respect to our new definition and has some additional properties such as being non-interactive, perfectly binding and reusable, which makes it of independent interest. This new definition allows us to prove soundness of the Dolev-Yao model extended with commitments, following the directions of Micciancio and Warinschi [MW04b].

Overview. Section 5.3 introduces basic notation and definitions from the literature. Section 5.4 elaborates on different definitions of non-malleability for commitment schemes and discusses the relations among them. In Section 5.5 we propose a new commitment scheme and we give a security proof. Section 5.2 describes symbolic protocol executions, its computational counterparts and the map between them and also states the soundness result. Finally in Section 5.7 there are some concluding remarks.

5.2 Symbolic Protocols

We are going to apply this theory to give sound computational interpretation to symbolic commitments. Recall from the introduction that the symbolic approach to protocol verification deals with symbolic or algebraic messages and idealized cryptographic primitives. In this setting the adversary is unbounded in running time and has full control over the communication media but is completely incapable of breaking the underlying cryptographic primitives.

We now describe the message space and the closure operator. These messages are used to formally describe cryptographic protocols. The closure represents the knowledge that can be extracted from a message, and it is used to define what valid algebraic protocol runs are. Intuitively a protocol run is valid if every message sent by a principal can be deduced from its knowledge except maybe for some fresh randomness. Much of this is standard (see, e.g., [AR02, MW04b, MP05, GvR06b]), except that we model commitments and decommitments as well as encryption.

Definition 5.2.1. Let **Nonce** be an infinite set of *nonce symbols*, **Const** a finite set of *constant symbols*, **Key** an infinite set of *key symbols*, and **Random** an infinite set of *randomness labels*. Nonces are denoted by $n, n', \ldots$, constants by $c, c', \ldots$, keys by $k, k', \ldots$, and randomness labels by $r, r', \ldots$. Using these building blocks, *messages* are constructed using symbolic encryption, commitments, decommitments, and pairing operations:

$$\mathbf{Msg} \ni m := c \mid n \mid \{\!|m|\!\}_k^r \mid \mathsf{com}^r(m) \mid \mathsf{dec}^r(m) \mid \langle m, m \rangle.$$

A message of the form $\{\!|m|\!\}_k^r$ is called an *encryption* and the set of all such messages is denoted by **Enc**. Similarly, messages of the form $\mathsf{com}^r(m)$ are called *commitments* and the set of all these messages is denoted by **Com**. The messages of the form $\mathsf{dec}^r(m)$ are called *decommitments* and the set of all these messages is denoted by **Dec**. In a protocol run $\mathsf{dec}^r(m)$ is a valid decommitment of $\mathsf{com}^{r'}(m')$ only if $m = m'$ and $r = r'$. We say that elements in $\mathbf{Const} \cup \mathbf{Nonce} \cup \mathbf{Key}$ are primitive and we denote this set by **Prim**. For a public key k we denote its associated private key as k^{-1}.

The *closure* of a set U of messages is the set of all messages that can be constructed from U using tupling, detupling, commitment, decommitment, and encryption and decryption. It represents the information an adversary could deduce knowing U. Note that, due to secrecy of the commitment scheme, knowing $\mathsf{com}^r(m)$ does not provide an adversary with any information about m.

Definition 5.2.2 (Closure)**.** Let U be a set of messages. The *closure* of U, denoted by $\overline{U}$, is the smallest set of messages satisfying:

1. $\mathbf{Const} \subseteq \overline{U}$;
2. $U \subseteq \overline{U}$;
3. $m, m' \in \overline{U} \implies \langle m, m' \rangle \in \overline{U}$;
4. $m \in \overline{U} \wedge k \in \overline{U} \implies \{\!|m|\!\}_k^r \in \overline{U}$;
5. $\{\!|m|\!\}_k^r \in \overline{U} \wedge k^{-1} \in \overline{U} \implies m \in \overline{U}$;
6. $m \in \overline{U} \implies \mathsf{com}^r(m), \mathsf{dec}^r(m) \in \overline{U}$;

7. $\mathsf{dec}^r(m) \in \overline{U} \implies m \in \overline{U}$;

8. $\langle m, m' \rangle \in \overline{U} \implies m, m' \in \overline{U}$.

Next we need to find the right security notions to give sound computational interpretation to symbolic encryption and commitments.

5.3 Computational Setup

This section introduces syntaxis and security definitions for different cryptographic primitives. Much of this is standard, we refer the reader to [GM84a, RS92] and [NY90] for a thorough explanation. Some of this primitives will be used to interpret algebraic operations and some of them are used as building blocks for our construction of Section 5.5.

5.3.1 Commitment Schemes

Definition 5.3.1. A *commitment scheme* is a triple $\Omega = (\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ of probabilistic polynomial-time algorithms. TTP, the *trusted third party*, takes as input the security parameter 1^η and produces a common reference string σ. We require that $|\sigma| \geq p(\eta)$ for some non-constant polynomial p. Snd, the *sender*, takes as input σ and a message m and produces a commitment *com* to this message and a corresponding decommitment *dec*. Rcv, the *receiver*, takes as input σ, *com*, and *dec* and produces a message or $\perp$.

Meaningfulness$_\Omega(A)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$m \leftarrow A(\sigma)$
$(com, dec) \leftarrow \mathsf{Snd}(\sigma, m)$
$m_1 \leftarrow \mathsf{Rcv}(\sigma, com, dec)$
return $m \neq m_1$

Secrecy$_{\mathsf{TTP},\mathsf{Snd}}(A_1, A_2)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$m_0, m_1, s \leftarrow A_1(\sigma)$
$b \leftarrow \{0, 1\}$
$(com, dec) \leftarrow \mathsf{Snd}(\sigma, m_b)$
$b' \leftarrow A_2(s, com)$
return $b = b'$

$\mathbf{Binding}_{\mathsf{TTP},\mathsf{Rcv}}(A)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$(com, dec_1, dec_2) \leftarrow A(\sigma)$
$m_1 \leftarrow \mathsf{Rcv}(\sigma, com, dec_1)$
$m_2 \leftarrow \mathsf{Rcv}(\sigma, com, dec_2)$
return $m_1 \neq \perp \neq m_2$
$\wedge\, m_1 \neq m_2$

The following three conditions must hold.

1. For all probabilistic polynomial-time algorithms A, the probability $\mathbb{P}[\mathbf{Meaningfullness}_\Omega(A)]$ is a negligible function of η.
2. For all probabilistic polynomial-time algorithms (A_1, A_2), the advantage
   $|\mathbb{P}[\mathbf{Secrecy}_{\mathsf{TTP},\mathsf{Snd}}(A_1, A_2)] - 1/2|$ is a negligible function of η.
3. For all probabilistic polynomial-time algorithms A, the probability $\mathbb{P}[\mathbf{Binding}_{\mathsf{TTP},\mathsf{Rcv}}(A)]$ is a negligible function of η.

Definition 5.3.2. A commitment scheme is said to be *perfectly binding* if for all unbounded algorithms A, the probability $\mathbb{P}[\mathbf{Binding}_{\mathsf{TTP},\mathsf{Rcv}}(A)]$ is zero.

Definition 5.3.3. A commitment scheme is said to be *perfectly hiding* if for all unbounded algorithms (A_0, A_1), $|\mathbb{P}[\mathbf{Secrecy}_{\mathsf{TTP},\mathsf{Snd}}(A_1, A_2)] - 1/2|$ is zero.

5.3.2 Encryption Schemes

Definition 5.3.4. An *encryption scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of probabilistic polynomial-time algorithms. $\mathcal{K}$ takes as input the security parameter 1^η and produces a key pair $(\mathrm{pk}, \mathrm{sk})$ where pk is the public encryption key and sk is the private decryption key. $\mathcal{E}$ takes as input a public key pk and a plaintext m and outputs a ciphertext. $\mathcal{D}$ takes as input a private key sk and a ciphertext and outputs a plaintext or $\perp$. It is required that $\mathbb{P}[(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}(1^\eta); c \leftarrow \mathcal{E}(\mathrm{pk}, m); m' \leftarrow \mathcal{D}(\mathrm{sk}, c) : m = m'] = 1$.

$$\begin{array}{l}\textbf{IND-CCA}_{\Pi}(A_0, A_1):\\ (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathcal{K}(1^\eta)\\ m_0, m_1, s \leftarrow A_0^{\mathcal{D}}(\mathrm{pk})\\ b \leftarrow \{0,1\}\\ c \leftarrow \mathcal{E}(\mathrm{pk}, m_b)\\ b' \leftarrow A_1^{\mathcal{D}}(s, c)\\ \textbf{return}\ \ b = b'\end{array}$$

Definition 5.3.5. An encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is said to be *IND-CCA secure* if for all probabilistic polynomial-time adversaries $A = (A_0, A_1)$ the advantage of A, defined as $|\mathbb{P}[\textbf{IND-CCA}_{\Pi}(A_0, A_1)] - 1/2|$, is a negligible function of η. This adversary has access to a decryption oracle $\mathcal{D}$ that on input c' outputs $\mathcal{D}(\mathrm{sk}, c')$ with the only restriction that $c \neq c'$.

5.3.3 One-time Signatures

Definition 5.3.6. A *signature scheme* is a triple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ of probabilistic polynomial-time algorithms. Gen takes as input the security parameter 1^η and produces a key pair $(\mathrm{vk}, \mathrm{sk})$ where vk is the signature verification key and sk is the secret signing key. Sign takes as input sk and a message m and produces a signature s of m. Vrfy takes as input vk, a message m and a signature s and outputs whether or not s is a valid signature of m.

$$\begin{array}{l}\textbf{OTS}_{\Sigma}(A_0, A_1):\\ (\mathrm{vk}, \mathrm{sk}) \leftarrow \mathsf{Gen}(1^\eta)\\ m, s \leftarrow A_0(\mathrm{vk}, 1^\eta)\\ \sigma \leftarrow \mathsf{Sign}(\mathrm{sk}, m)\\ m', \sigma' \leftarrow A_1(s, \sigma)\\ \textbf{return}\ \ \sigma \neq \sigma' \wedge \mathsf{Vrfy}(\mathrm{vk}, (m', \sigma'))\end{array}$$

Definition 5.3.7. A signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a *strong, one-time signature scheme* if the success probability of any probabilistic polynomial-time adversary (A_0, A_1) in the game $\textbf{OTS}_{\Sigma}(A_0, A_1)$ is negligible in the security parameter η.

5.3.4 Tag-based Encryption

Definition 5.3.8. A *tag-based encryption scheme (TBE)* handling tags of length ℓ (where ℓ is a polynomially-bounded function) is a triple of probabilistic polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. KeyGen takes a security parameter 1^η and returns a public key pk and secret key sk. The public key pk includes the security parameter 1^η and $\ell(\eta)$; as well as the description of sets $\mathcal{M}, \mathcal{R}, \mathcal{C}$, which denote the set of messages, randomness and ciphertexts respectively. These descriptions might depend on the public key pk. Enc takes as inputs pk, a tag $t \in \{0,1\}^\ell$ and $m \in \mathcal{M}$. It returns a ciphertext $c \in \mathcal{C}$. Dec takes as inputs the secret key sk, a tag t and $c \in \mathcal{C}$, and returns $m \in \mathcal{M}$ or $\perp$ when c is not a legitimate ciphertext. For the sake of consistency, these algorithms must satisfy $\mathsf{Dec}(\mathrm{sk}, t, c) = m$ for all $t \in \{0,1\}^\ell,\ m \in \mathcal{M}$, where $c = \mathsf{Enc}(\mathrm{pk}, t, m)$.

Definition 5.3.9. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a TBE scheme. We say $\mathcal{E}$ is *IND-TBE-CCA secure* if for any 3-tuple of PPT oracle algorithms (A_0, A_1, A_2) and any polynomially-bounded function ℓ the advantage in the following game is negligible in the security parameter 1^η:

$A_0(1^\eta, \ell(\eta))$ outputs a target tag t. $\mathsf{KeyGen}(1^\eta)$ outputs $(\mathrm{pk}, \mathrm{sk})$ and the adversary is given pk. Then the adversary A_1 may ask polynomially-many queries to a decryption oracle $\mathcal{D}(t', c') = \mathsf{Dec}(\mathrm{sk}, t', c')$ for pairs tag-ciphertext (t', c') of its choice, with the restriction $t \neq t'$. At some point, A_1 outputs two equal length messages m_0, m_1. A bit $b \leftarrow \{0,1\}$ is chosen at random and the adversary is given a challenge ciphertext $c \leftarrow \mathsf{Enc}(\mathrm{pk}, t, m_b)$. A_2 may continue asking the decryption oracle for pairs tag-ciphertext (t', c') of its choice, with the restriction $t \neq t'$. Finally, A_2 outputs a guess b'.

IND-TBE-CCA $_{\mathcal{E}}(A_0, A_1, A_2)$:
$t, s_1 \leftarrow A_0(1^\eta, \ell(\eta))$
$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{KeyGen}(1^\eta)$
$m_0, m_1, s_2 \leftarrow A_1^{\mathcal{P}}(s_1, \mathrm{pk})$
$b \leftarrow \{0, 1\}$
$c \leftarrow \mathsf{Enc}(\mathrm{pk}, t, m_b)$
$b' \leftarrow A_2^{\mathcal{P}}(s_2, c)$
return $b = b'$

We define the advantage of A as $|\mathbb{P}[\mathbf{IND\text{-}TBE\text{-}CCA}(A)] - 1/2|$.

5.3.5 Interpretation

Suppose we have an encryption scheme Π, a commitment scheme Ω and a function that maps symbolic constants to constant bitstrings. Then we can define a mapping $[\![\cdot]\!]$ from algebraic messages $m \in \mathbf{Msg}$ to distributions over bitstrings $[\![m]\!] \in \mathbf{Str}$. This interpretation maps nonces to random bitstrings of length η; encryptions are interpreted by running the encryption algorithm $\mathcal{E}$ and for interpreting commitments and decommitments we use the commit algorithm Snd.

In order to achieve sound interpretation we will explore the security requirements on these cryptographic primitives. For the case of encryption it is satisfactory to use any IND-CCA encryption scheme as shown in [MW04b]. For the case of commitments, using standard security definitions is not straightforward as they are not strong enough nor indistinguishability based. To achieve sound interpretation of the idealized Dolev-Yao model, throughout the next section we elaborate on a convenient security definition for commitment schemes.

5.4 Definitions of Non-malleability

As noticed by Fischlin and Fischlin [FF00], there are two different versions of non-malleability for commitment schemes, namely: NM with respect to opening (NMO) and NM with respect to commitment (NMC). NMC was

the version originally proposed by Dolev, Dwork and Naor in [DDN91]. It means that given a commitment to a message m, the adversary is unable to build a different commitment to m', with m related to m'. This version of non-malleability is appropriate while considering perfectly binding commitments and only makes sense for schemes that are not perfectly hiding.

The other version NMO, seemingly weaker, means that an adversary that is first given a commitment to m and on a second stage its decommitment, is unable to find a different commitment-decommitment pair that decommits to a message m' related to m. This notion was studied by Di Crescenzo, Ishai and Ostrovsky [CIO98] and later by Di Crescenzo, Katz, Ostrovsky and Smith [CKOS01]. Intuitively a commitment scheme is non-malleable if the adversary can do no better than a simulator which has no information at all about the message that was committed to. Next we recall their definition.

$\mathbf{NMO}_{\Omega}(A_1, A_2, D, R)$:
$\sigma \leftarrow \mathsf{TTP}(1^{\eta})$
$m_1 \leftarrow D$
$com_1, dec_1 \leftarrow \mathsf{Snd}(\sigma, m_1)$
$com_2 \leftarrow A_1(\sigma, com_1)$
$dec_2 \leftarrow A_2(\sigma, com_1, com_2, dec_1)$
$m_2 \leftarrow \mathsf{Rcv}(\sigma, com_2, dec_2)$
return $com_1 \neq com_2 \wedge R(m_1, m_2)$

$\mathbf{SIM}(S, D, R)$:
$m_1 \leftarrow D$
$m_2 \leftarrow S(1^{\eta}, D)$
return $R(m_1, m_2)$

Definition 5.4.1 (Non-malleability [CIO98, CKOS01])**.** Let $\Omega = (\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ be a commitment scheme. Ω is called *non-malleable* if for all PPT adversaries (A_1, A_2) there is a PPT simulator S such that for all distributions D and all relations R,

$$\mathbb{P}[\mathbf{NMO}_{\Omega}(A_1, A_2, D, R)] - \mathbb{P}[\mathbf{SIM}(S, D, R)]$$

is a negligible function of η.

Remark 5.4.1. To prevent that the adversary trivially wins, by refusing to decommit, the following restriction over the relation R is imposed: for all messages m, we have $R(m, \perp) = 0$.

5.4.1 NMC-CCA: Non-malleability Against Chosen Commitment Attacks

The previous definition deals with non-malleability with respect to opening. For the relation between symbolic and computational cryptography we need the stronger notion of non-malleability with respect to commitment. Intuitively, this is because in the algebraic setting $\mathsf{com}^r(m')$ cannot be deduced from $\mathsf{com}^r(m)$, with m' somehow related to m. Therefore we adapt the NMO definition to non-malleability with respect to commitment and we strengthen it by incorporating active adaptive security, allowing the adversary to mount *chosen commitment attacks* (CCA in short). Specifically, we empower the adversary with access to a decommitment oracle $\mathcal{D}$. To do so, from now on, we restrict our attention to non-interactive, perfectly binding trapdoor commitment schemes. The oracle $\mathcal{D}$ has access to the trapdoor information. It takes as argument a commitment c with the restriction that c is not equal to the challenge commitment com_1. Then if the commitment c has been correctly generated, the oracle returns a decommitment d which opens c, and otherwise it outputs $\perp$.

NMC-CCA$_\Omega(A_0, A_1, R)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$D, s_1 \leftarrow A_0^{\mathcal{D}}(\sigma)$
$m_1 \leftarrow D(\sigma)$
$com_1, dec_1 \leftarrow \mathsf{Snd}(\sigma, m_1)$
$com_2, s_r \leftarrow A_1^{\mathcal{D}}(s_1, com_1)$
$dec_2 \leftarrow \mathcal{D}(com_2)$
$m_2 \leftarrow \mathsf{Rcv}(\sigma, com_2, dec_2)$
return $com_1 \neq com_2 \wedge R(s_r, m_1, m_2)$

SIM-CCA$_{\mathsf{TTP}}(S_0, S_1, R)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$D, s_1 \leftarrow S_0(\sigma)$
$m_1 \leftarrow D(\sigma)$

$com_2, s_r \leftarrow S_1(s_1)$
$dec_2 \leftarrow \mathcal{D}(com_2)$
$m_2 \leftarrow \mathsf{Rcv}(\sigma, com_2, dec_2)$
return $R(s_r, m_1, m_2)$

Definition 5.4.2 (NMC-CCA)**.** Let $\Omega = (\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ be a commitment scheme. Ω is called *NMC-CCA secure* if for all PPT adversaries (A_0, A_1) there is a PPT simulator (S_0, S_1) such that for all relations R (with the same restriction as in 5.4.1),

$$\mathbb{P}[\mathbf{NMC\text{-}CCA}_\Omega(A_0, A_1, R)] - \mathbb{P}[\mathbf{SIM\text{-}CCA}_{\mathsf{TTP}}(S_0, S_1, R)]$$

is a negligible function of η.

5.4.2 An Indistinguishability Based Definition

Next we introduce an equivalent formulation of NMC-CCA that is more convenient to prove soundness of the Dolev-Yao model with respect to commitment schemes.

IND-COM-CCA$_b(A_0, A_1)$:
$\sigma \leftarrow \mathsf{TTP}(1^\eta)$
$m_0, m_1, s_1 \leftarrow A_0^{\mathcal{D}}(\sigma)$
$com_1, dec_1 \leftarrow Snd(\sigma, m_b)$
$b' \leftarrow A_1^{\mathcal{D}}(s_1, com_1)$
return b'

Definition 5.4.3 (IND-COM-CCA)**.** Let $\Omega = (\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ be a commitment scheme. Ω is said to be *IND-COM-CCA secure* if for all PPT adversaries (A_0, A_1)

$$\mathbb{P}[\textbf{IND-COM-CCA}_1(A_0^{\mathcal{D}}, A_1^{\mathcal{D}}) = 1] - \mathbb{P}[\textbf{IND-COM-CCA}_0(A_0^{\mathcal{D}}, A_1^{\mathcal{D}}) = 1]$$

is a negligible function of η.

Next we show that NMC-CCA and IND-COM-CCA are equivalent. We discuss it briefly as it is basically the proof that NM-CCA and IND-CCA are equivalent, adapted to commitment schemes.

Theorem 5.4.4. *Let $\Omega = (\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ be a commitment scheme. Then Ω is IND-COM-CCA secure if and only if Ω is NMC-CCA secure.*

Proof. (IND-COM-CCA $\Leftarrow$ NMC-CCA) Let (B_0, B_1) be an adversary for IND-COM-CCA. Then we build the following adversary (A_0, A_1) against NMC-CCA.

Algorithm $A_0^{\mathcal{D}}(\sigma)$:
$m_0, m_1, s_1 \leftarrow B_0^{\mathcal{D}}(\sigma)$
$D \leftarrow U(\{m_0, m_1\})$
return $D, (\sigma, m_0, m_1, s_1)$

Algorithm $A_1^{\mathcal{D}}((\sigma, m_0, m_1, s_1), c_1)$:
$b \leftarrow B_1^{\mathcal{D}}(s_1, c_1)$
$c_2 \leftarrow \mathsf{Snd}(\sigma, m_b)$
return c_2, ϵ

where U is the uniform distribution. Now take the relation $R(s_r, m_1, m_2)$ as m_1 equal to m_2. It should be clear, after unfolding (A_0, A_1) in the **NMC-CCA** game, that this adversary has the same advantage that (B_0, B_1) has against IND-COM-CCA.

(IND-COM-CCA $\Rightarrow$ NMC-CCA) Let (A_0, A_1) be an adversary for NMC-CCA. Then we build the following adversary (B_0, B_1) against IND-COM-CCA.

Algorithm $B_0^{\mathcal{D}}(\sigma)$:
$D, s_1 \leftarrow A_0^{\mathcal{D}}(\sigma)$
$m_0, m_1 \leftarrow D$
return $m_0, m_1, (\sigma, m_0, m_1, s_1)$

Algorithm $B_1^{\mathcal{D}}((\sigma, m_0, m_1, s_1), c_1)$:
$c_2 \leftarrow A_1^{\mathcal{D}}(s_1, c_1)$
$m \leftarrow \mathcal{D}(c_2)$
if $m = m_1$ **then return** 1
else return 0

Again, just by unfolding these adversaries in the IND-COM-CCA game, it is easy to verify that they have the same advantage that (A_0, A_1) has against NMC-CCA. □

It remains to show that such a security notion for a commitment scheme is achievable. In the next section we give a practical construction that achieves IND-COM-CCA security.

5.5 The Construction

We now propose a new construction for IND-COM-CCA that is computationally hiding, perfectly binding, reusable, non-interactive, non-malleable under adaptive adversaries, and provably secure under the assumption that trapdoor permutations exist.

Next we outline the idea of our construction. As pointed out by Di Crescenzo, Katz, Ostrovsky and Smith [CKOS01], an IND-CCA secure public key encryption scheme can be converted into a perfectly binding non-malleable commitment scheme. Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an indistinguishable against adaptive chosen-ciphertext attacks secure public key encryption scheme. The idea is to commit to a message m by encrypting it using random coins r; commitment is set to be the ciphertext $c =$

$\mathsf{Enc}(\mathrm{pk}, m; r)$; de-commitment is set to be the pair (m, r); finally the opening algorithm takes (c, m, r) and checks whether $c = \mathsf{Enc}(\mathrm{pk}, m; r)$. When trying to directly use this construction to instantiate an IND-COM-CCA commitment scheme one might not be able to simulate the de-commitment oracle. The reason is that given a ciphertext/commitment c, one recovers the purported embedded message m by using the decryption algorithm, but not necessarily the randomness r. One way to break through this situation is to include in the commitment a second ciphertext $c' = \mathsf{Enc}(\mathrm{pk}', r; r')$ encrypting the randomness r used in the first ciphertext $c = \mathsf{Enc}(\mathrm{pk}, m; r)$. This is the key idea of our construction. We additionally use one-time signatures and this together with tag-based encryption schemes ensure the de-commitment oracle does not leak vital information.

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a tag based encryption scheme and let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. Define $(\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ as follows:

- TTP runs $\mathsf{KeyGen}(1^\eta)$ twice to obtain $(\mathrm{pk}_1, \mathrm{sk}_1)$ and $(\mathrm{pk}_2, \mathrm{sk}_2)$. The common reference string includes $\mathrm{pk}_1, \mathrm{pk}_2$.
- To commit to a message m, the sender Snd computes and outputs the commitment $C = (\mathrm{vk}, c_1, c_2, s)$ where $c_1 = \mathsf{Enc}(\mathrm{pk}_1, \mathrm{vk}, m; r_1)$, $c_2 = \mathsf{Enc}(\mathrm{pk}_2, \mathrm{vk}, r_1; r_2)$, with $r_1, r_2 \leftarrow \mathcal{R}$, $(\mathrm{vk}, \mathrm{sk}) \leftarrow \mathsf{Gen}(1^\eta)$ and $s \leftarrow \mathsf{Sign}(\mathrm{sk}, (c_1, c_2))$. The decommitment is set to be (m, r_1).
- To de-commit ciphertext $C = (\mathrm{vk}, c_1, c_2, s)$ using (m, r_1), the receiver Rcv first checks if the signature on (c_1, c_2) is correct, and afterwards whether or not $c_1 = \mathsf{Enc}(\mathrm{pk}_1, \mathrm{vk}, m; r_1)$.

We assume $\mathcal{R} = \mathcal{M}$.

Theorem 5.5.1. *Assume that* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is an IND-TBE-CCA secure tag based encryption scheme and that* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a one-time strongly unforgeable signature scheme. Then* $(\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ *is an IND-COM-CCA secure commitment scheme.*

Proof. We transform an adversary A against the IND-COM-CCA security of the commitment scheme into adversaries against the TBE and the OTS.

Next we will describe a sequence of games following the methodology advocated in [Sho04, BR06]. Let X_i be the event that A learns the challenge bit b in the i-th game.

Game 0. This is the unmodified IND-COM-CCA game. Trivially, $|\mathbb{P}[X_0] - 1/2|$ equals the advantage of A against IND-COM-CCA.

Game 1. In this game we disallow decryption queries $C = (\mathrm{vk}, c_1, c_2, s)$ s.t. $\mathrm{vk} = \mathrm{vk}^\star$ where $(\mathrm{vk}^\star, c_1^\star, c_2^\star, s^\star)$ is the challenge commitment. Then, we get that $|\mathbb{P}[X_1] - \mathbb{P}[X_0]|$ is less or equal than the advantage any PPT algorithm has in breaking the one-time strong unforgeability security of the OTS.

Game 2. Still decryption queries with $\mathrm{vk} = \mathrm{vk}^\star$ are forbidden. In this game we use the IND-CCA security of the second instance of the TBE scheme. The components $c_1^\star$ and $c_2^\star$ of the challenge ciphertext are changed to $c_1^\star = \mathsf{Enc}(\mathrm{pk}_1^\star, \mathrm{vk}^\star, m_b^\star; r_1)$, and $c_2^\star = \mathsf{Enc}(\mathrm{pk}_2^\star, \mathrm{vk}^\star, r')$ where $r', r_1 \leftarrow \mathcal{R}$. Now, we have $|\mathbb{P}[X_2] - \mathbb{P}[X_1]|$ is less or equal than the advantage any PPT algorithm has in breaking the selective IND-CCA security of the TBE.

Finally it is shown that $|\mathbb{P}[X_2] - 1/2]|$ is bounded by the advantage any PPT algorithm has in breaking the selective IND-CCA security of the first instance of the TBE.

Putting everything together, we get that $|\mathbb{P}[X_0] - 1/2|$ is bounded by the advantages in breaking the OTS scheme plus twice the advantage in breaking the selective IND-CCA of the TBE scheme. Next we describe the concrete adversaries,

Game 0 $\approx$ Game 1. Assume that there is an adversary (A_0, A_1) that is able to distinguish the environments of Game 0 and 1. Then we build an adversary (B_0, B_1) against the one-time strong unforgeability of the signature scheme.

Algorithm $B_0(1^\eta, \mathrm{vk})$:
$\mathrm{pk}_1, \mathrm{sk}_1 \leftarrow \mathsf{KeyGen}(1^\eta)$
$\mathrm{pk}_2, \mathrm{sk}_2 \leftarrow \mathsf{KeyGen}(1^\eta)$
$m_0, m_1, s_1 \leftarrow A_0^{\mathcal{D}}(\mathrm{pk}_1, \mathrm{pk}_2)$
$b \leftarrow \{0,1\}$
$r_1 \leftarrow \mathcal{R}$
$c_1 \leftarrow \mathsf{Enc}(\mathrm{pk}_1, \mathrm{vk}, m_b; r_1)$
$c_2 \leftarrow \mathsf{Enc}(\mathrm{pk}_2, \mathrm{vk}, r_1)$
return $(c_1, c_2), (s_1||\mathrm{vk}||c_1||c_2||\mathrm{sk}_1||\mathrm{sk}_2)$

and $B_1((s_1||\mathrm{vk}||c_1||c_2||\mathrm{sk}_1||\mathrm{sk}_2), s) = [b' \leftarrow A_1^{\mathcal{D}}(s_1, (\mathrm{vk}, c_1, c_2, s))]$. Calls to the decommitment oracle $\mathcal{D}(\mathrm{vk}', c_1', c_2', s')$ are simulated by firstly verifying the signature $\mathsf{Vrfy}(\mathrm{vk}', (c_1', c_2'), s')$. If the verification succeeds then the oracle returns the pair $(\mathsf{Dec}(\mathrm{sk}_1, \mathrm{vk}', c_1'), \mathsf{Dec}(\mathrm{sk}_2, \mathrm{vk}', c_2'))$ and otherwise it outputs $\perp$. If the adversary eventually performs a query $\mathcal{D}(\mathrm{vk}', c_1', c_2', s')$ with $\mathrm{vk}' = \mathrm{vk}$ then the execution of the adversary is aborted and B outputs $((c_1', c_2'), s')$, thus breaking the one-time strong unforgeability of the signature scheme.

Game 1 $\approx$ Game 2. Assume that there is an adversary (A_0, A_1) that is able to distinguish the environments of Game 1 and 2. Then we build an adversary (B_0, B_1, B_2) against the IND-CCA security of the second TBE. Take $B_0(1^\eta, \ell(\eta)) = [(\mathrm{vk}, \mathrm{sk}) \leftarrow \mathsf{Gen}(1^\eta); \mathbf{return}\ \mathrm{vk}, (\mathrm{vk}||\mathrm{sk})]$ and $B_1(s_1, \mathrm{pk}_2) = [r', r_1 \leftarrow \mathcal{R}; \mathbf{return}\ r', r_1, (s_1||r'||r_1||\mathrm{pk}_2)]$ and

Algorithm $B_2^{\mathcal{O}_{\mathrm{sk}_2}}((\mathrm{vk}||\mathrm{sk}||r'||r_1||\mathrm{pk}_2), c_2)$:
$\mathrm{pk}_1, \mathrm{sk}_1 \leftarrow \mathsf{KeyGen}(1^\eta)$
$m_0, m_1, s_1 \leftarrow A_0^{\mathcal{D}}(\mathrm{pk}_1, \mathrm{pk}_2)$
$b \leftarrow \{0,1\}$
$c_1 \leftarrow \mathsf{Enc}(\mathrm{pk}_1, \mathrm{vk}, m_b; r_1)$
$s \leftarrow \mathsf{Sign}(\mathrm{sk}, (c_1, c_2))$
$b' \leftarrow A_1^{\mathcal{D}}(s_1, (\mathrm{vk}, c_1, c_2, s))$
if $b = b'$ **then return** 1
else return 0

Calls to the decommitment oracle $\mathcal{D}(\mathrm{vk}, c_1, c_2, s)$ are simulated by firstly verifying the signature $\mathsf{Vrfy}(\mathrm{vk}, (c_1, c_2), s)$. If the verification succeeds then

the oracle returns $(\mathsf{Dec}(\mathrm{sk}_1, \mathrm{vk}, c_1), \mathcal{O}_{\mathrm{sk}_2}(c_2))$ and otherwise it outputs $\perp$.

Finally we show that $|\mathbb{P}[X_2] - 1/2]|$ is bounded by the advantage any PPT algorithm has in breaking the selective IND-CCA security of the first instance of the TBE. Assume that there is an adversary (A_0, A_1) for Game 2. Then we build an adversary (B_0, B_1, B_2) against the IND-CCA security of the first TBE scheme. Take $B_0(1^\eta, \ell(\eta)) = [(\mathrm{vk}, \mathrm{sk}) \leftarrow \mathsf{Gen}(1^\eta); \textbf{return } \mathrm{vk}, (\mathrm{vk}||\mathrm{sk})]$ and

Algorithm $B_1^{\mathcal{O}_{\mathrm{sk}_1}}((\mathrm{vk}||\mathrm{sk}), \mathrm{pk}_1)$:
$\mathrm{pk}_2, \mathrm{sk}_2 \leftarrow \mathsf{KeyGen}(1^\eta)$
$m_0, m_1, s_1 \leftarrow A_0^{\mathcal{D}}(\mathrm{pk}_1, \mathrm{pk}_2)$
return $(m_0, m_1), (\mathrm{vk}||\mathrm{sk}||\mathrm{pk}_1||s_1||\mathrm{pk}_2||\mathrm{sk}_2)$

Algorithm $B_2^{\mathcal{O}_{\mathrm{sk}_1}}((\mathrm{vk}||\mathrm{sk}||\mathrm{pk}_1||s_1||\mathrm{pk}_2||\mathrm{sk}_2), c_1)$:
$r' \leftarrow \mathcal{R}$
$c_2 \leftarrow \mathsf{Enc}(\mathrm{pk}_2, \mathrm{vk}, r')$
$s \leftarrow \mathsf{Sign}(\mathrm{sk}, (c_1, c_2))$
$b' \leftarrow A_1^{\mathcal{D}}(s_1, (\mathrm{vk}, c_1, c_2, s))$
return b'

Calls to the decommitment oracle $\mathcal{D}(\mathrm{vk}, c_1, c_2, s)$ are simulated by firstly verifying the signature $\mathsf{Vrfy}(\mathrm{vk}, (c_1, c_2), s)$. If the verification succeeds then the oracle returns $(\mathcal{O}_{\mathrm{sk}_1}(c_1), \mathsf{Dec}(\mathrm{sk}_2, \mathrm{vk}, c_2))$ and otherwise it outputs $\perp$. □

5.6 Protocol Execution and State Traces

We now prove that it is possible to port proofs in the symbolic framework to the computational one. First, for the sake of self-containment we describe the adversarial model and the execution environment following the directions of Micciancio and Warinschi [MW04b]. We refer the reader to this paper for a thorough explanation.

The message space and the closure operator were defined in Section 5.2. Messages are used to formally describe cryptographic protocols. The closure represents the knowledge that can be extracted from a message, and

is used to define what valid algebraic protocol runs are. Intuitively a protocol run is valid if every message sent by a principal can be deduced from its knowledge except maybe for some fresh randomness. In this setting an adversary is in control of the communication media and is able to interact with honest participants. Consider then an adversary that has access to an oracle that will play the role of the honest participants. This adversary can start new sessions of the protocol and send messages to a principal of a given session and get the respective answer back. Formally, the adversary A can perform one of the following queries to the execution oracle $\mathcal{O}$.

1. $\mathsf{newsession}([I_1 \dots I_n])$ that takes a list of user identities I_i and returns a new session identifier s.
2. $\mathsf{send}(s, I, m)$ that delivers the message m to the principal I of session s. Then $\mathcal{O}$ updates I's state and returns the answer to the adversary.

In case that the adversary performs a query that is not according to the protocol, for the specific state of the receiver, the oracle aborts the execution of this session.

In a formal protocol, the messages exchanged are algebraic expressions from the message algebra. A formal adversary A^f will interact with the formal oracle $\mathcal{O}^f$ in a symbolic protocol run.

On the other hand, a computational adversary A^c is a probabilistic polynomial-time Turing machine that operates on bitstrings. For a fixed value of the security parameter there is a set of primitive bitstrings for constants and nonces denoted by $\mathbf{Prim}_\eta$. The set of bitstrings $\mathbf{Msg}_\eta$ is build from $\mathbf{Prim}_\eta$ by tupling, encryptions, commitments and decommitments. There is a set **Sid** of *session identifiers*; a set **Uid** of *user identities* and a set **Vars** of *variables* in the abstract protocol description.

Let $F : \mathbf{Sid} \times \mathbf{Uid} \to (\mathbf{Vars} \to \mathbf{Msg}, \mathbb{N})$ be the state maintained by the formal oracle $\mathcal{O}^f$. On input (s, I) it returns the state of principal I in session s together with his *instruction pointer*. The instruction pointer indicates on which step of the abstract protocol this principal is. Similarly, $C : \mathbf{Sid} \times \mathbf{Uid} \to (\mathbf{Vars} \to \mathbf{Msg}_\eta, \mathbb{N})$ is the state maintained by the computational oracle $\mathcal{O}^c$. Assume without loss of generality that all the sessions

are created at the beginning. Then, a formal adversary A_f is just a sequence of $\mathsf{send}(s, I, m)$ queries. We say that a formal adversary A_f is a valid Dolev-Yao adversary ($A_f \in \mathbf{DY}$) if each message he sends to the oracle is in the closure of his initial knowledge plus the answers he gets from the oracle $\mathcal{O}^f$. A protocol execution, thus, is the sequence of states $F_0, F_1, \ldots$ of the formal oracle O^f and is denoted by $\mathsf{trace}(A_f, \mathcal{O}^f)$. After fixing the randomness of the adversary and that of the oracle environment to τ_A and $\tau_{\mathcal{O}}$, we can similarly define a computational execution trace $\mathsf{trace}(A_c(\tau_A), \mathcal{O}^c(\tau_{\mathcal{O}}))$ as the sequence of states $C_0, C_1, \ldots$ of the computational oracle $\mathcal{O}^c$.

Definition 5.6.1. We say that $[\![\cdot]\!]\colon \mathbf{Prim} \to \mathbf{Prim}_\eta$ is an *interpretation function* if it is injective and structure preserving (i.e., maps formal nonces to nonce bitstrings, formal commitments to commitments and so on).

Definition 5.6.2. Let $F = F_0, F_1, \ldots$ be a formal execution trace and let $C = C_0, C_1, \ldots$ be a concrete execution trace. We say that $F \preceq C$ if there exists an interpretation function $[\![\cdot]\!]$ such that $[\![F_0]\!] = C_0, [\![F_1]\!] = C_1, \ldots$.

The following theorem shows that a computational adversary has no more power than an algebraic adversary.

Theorem 5.6.3. *Let* $(\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ *be an IND-COM-CCA secure commitment scheme and let* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an IND-CCA secure encryption scheme. For any computational adversary* A_c*, the probability*

$$\mathbb{P}[\,\exists A_f \in \boldsymbol{DY} : \mathsf{trace}(A_f, \mathcal{O}^f) \preceq \mathsf{trace}(A_c(\tau_A), \mathcal{O}^c(\tau_{\mathcal{O}}))]$$

is overwhelming. Here the probability is taken over the random choices τ_A *of the adversary and* $\tau_{\mathcal{O}}$ *of the oracle.*

Proof. First fix the randomness τ_A and $\tau_{\mathcal{O}}$. Running the computational adversary A_c, it produces a sequence of queries/answers to/from the computational oracle. Because we know all the trapdoor information that the oracle generates and because the adversary has to send properly typed messages, we can de-construct any message sent into primitive terms. Choosing new algebraic terms for each distinct primitive bitstring encountered we build a

sequence of algebraic queries which constitutes an algebraic adversary A_f. Note that for different random choices of τ_A and $\tau_{\mathcal{O}}$ we get the same A_f (up to renaming) with overwhelming probability.

It remains to show that the adversary we just built is Dolev-Yao. Suppose that it is not. Then A_f must, at some point, send a query that contains a non-adversarial nonce $n^\star$ that is not in the closure of the messages he received before. If this nonce occurs inside an encryption (with an unknown key) then one can build an adversary breaking the IND-CCA security of the encryption scheme [MW04b]. Assume then that it occurs inside a commitment. We now build an adversary that breaks the IND-COM-CCA security of the commitment scheme.

This adversary simulates the environment to A_c using the de-commit oracle when necessary except for the query that contains $n^\star$. There it generates two interpretations (n_0, n_1) for $n^\star$ and gives them as challenge plaintext for the IND-COM-CCA game. The challenger gives back a commitment to n_b where b is the challenge bit. This commitment to n_b is used to answer the oracle queries. At the moment A_C outputs the interpretation of $n^\star$ we can check whether it is n_0 or n_1. □

A formal security notion is a predicate $\mathsf{P_f}$ on formal traces. A protocol $\Pi \models_f \mathsf{P_f}$ if for all adversaries $A_f \in \mathbf{DY}$ holds that $\mathsf{trace}(A_f, \mathcal{O}^f) \in \mathsf{P_f}$. Similarly, a computational security notion is a predicate $\mathsf{P_c}$ on computational traces. A protocol $\Pi \models_c \mathsf{P_c}$ if for all probabilistic polynomial-time adversaries A_c holds that $\mathsf{trace}(A_f, \mathcal{O}^f) \in \mathsf{P_c}$ with overwhelming probability (taken over the random choices of the adversary and the ones of the oracle environment). The proof of the following theorem follows as in [MW04b].

Theorem 5.6.4. *Let* $(\mathsf{TTP}, \mathsf{Snd}, \mathsf{Rcv})$ *be a IND-COM-CCA secure commitment scheme and let* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an IND-CCA secure encryption scheme. Let* $\mathsf{P_f}$ *and* $\mathsf{P_c}$ *be respectively formal and computational security notions such that for all formal traces ft and all computational traces ct it holds that* $(ft \in \mathsf{P_f} \wedge ft \preceq ct) \Longrightarrow ct \in \mathsf{P_c}$. *Then*

$$\Pi \models_f \mathsf{P_f} \implies \Pi \models_c \mathsf{P_c} .$$

□

5.7 Conclusions

We presented two equivalent security notions for commitment schemes: a simulation based definition and a indistinguishability based one. We then gave a concrete scheme satisfying this security notion. This construction is of interest on itself as it is generic and has some interesting features like being reusable, perfectly binding and secure against adaptive chosen-commitment attacks. We then applied this new machinery to give sound interpretation of symbolic commitments while considering active adversaries.

Bibliography

[ABHS05] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness of formal encryption in the presence of key-cycles. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396. Springer Verlag, 2005.

[ABMW05] Martín Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Transactions on Internet Technology*, 5(2):299–327, 2005.

[ABS05] Pedro Adão, Gergei Bana, and Andre Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proceedings of the 18th IEEE Computer Security Foundations Workshop, (CSFW'05)*, pages 170–184. IEEE, 2005.

[ABW06] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In Luca Aceto and Anna Ingólfsdóttir, editors, *9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer Verlag, 2006.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM Conference of Computer and Communication Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[AH00] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000. `http://firstmonday.org/issues/issue5_10/adar/index.html`.

[AJ01] Martín Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation. In Naoki Kobayashi and Benjamin C. Pierce, editors, *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software (TACS'01)*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer Verlag, 2001.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[AW05] Martín Abadi and Bogdan Warinschi. Security analysis of cryptographically controlled access to XML documents. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 108–117. ACM Press, 2005.

[Bac97] Adam Back. Hashcash - a denial of service counter-measure. `http://www.cypherspace.org/hashcash`, 1997.

[BCK05] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer Verlag, 2005.

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Philip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 394–405. IEEE, 1997.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[BHF03] Fran Berman, Anthony J. G. Hey, and Geoffrey C. Fox. *Grid Computing: Making The Global Infrastructure a Reality*. Wiley, 2003.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proceedings of the 14th IEEE workshop on Computer Security Foundations (CSFW'01)*, pages 82–96. IEEE, 2001.

[BLMW07] Emmanuel Bresson, Yassine Lakhnech, Laurent Mazaré, and Bogdan Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In A. J. Menezes, editor, *Proceedings the IACR International Conference: Advances in Cryptology (CRYPTO'07)*, Lecture Notes in Computer Science. Springer Verlag, 2007.

[BPW06] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Limits of the BRSIM/UC soundness of dolev-yao models with hashes. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, pages 404–423. Springer Verlag, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceed-*

ings of the 1st ACM Conference on Computer and Communication Security (CCS'93), pages 62–73. ACM Press, 1993.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology (EUROCRYPT'06)*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer Verlag, 2006.

[Can97a] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burt Kaliski, editor, *Advances in Cryptology (CRYPTO'97)*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer Verlag, 1997.

[Can97b] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. Cryptology ePrint Archive, Report 1997/007 (`http://eprint.iacr.org/1997/007`), 1997.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science (FOCS'01)*, page 136. IEEE, 2001.

[CDG+02] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *ACM SIGOPS Operating Systems Review*, 36(SI):299–314, 2002.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Chv79] V. Chvtal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.

[CIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 141–150. ACM Press, 1998.

[CKKW06] Véronique Cortier, Steve Kremer, Ralf Küsters, and Bogdan Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In Naveen Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187. Springer Verlag, 2006.

[CKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In B. Pfitzmann, editor, *Advances in Cryptology (EUROCRYPT'01)*, volume 2045, pages 40–59. Springer Verlag, 2001.

[CMR98] Ran Canetti, Danielle Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 131–140. ACM Press, 1998.

[Coh03] B. Cohen. Incentives build robustness in bittorrent. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[CP92] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In R. A. Rueppel, editor, *Advances in Cryptology (EUROCRYPT'92)*, volume 658 of *Lecture Notes in Computer Science*, pages 390–407. Springer Verlag, 1992.

[Cre06] C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. PhD thesis, Eindhoven University of Technology, 2006.

[CS03a] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key enscryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33:167–226, 2003.

[CS03b] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer Verlag, 2003.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *International workshop on Designing privacy enhancing technologies*, pages 46–66. Springer Verlag, 2001.

[Dam88] Ivan Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology (EUROCRYPT'87)*, pages 203–216. Springer Verlag, 1988.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)*, pages 2–15, 2003.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 542–552. ACM Press, 1991.

[DGN02] Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In Dan Boneh, editor, *Advances in Cryptology (CRYPTO'03)*, volume 2729 of *Lecture Notes in Computer Science*, pages 426–444. International Association for Cryptologic Research, Springer Verlag, 2002.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium (SSYM'04)*, pages 21–21. USENIX Association, 2004.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, 1981.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomizing protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[FF00] Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In *Advances in Cryptology (CRYPTO'03)*, volume 1880 of *Lecture Notes in Computer Science*, pages 413–431. Springer Verlag, 2000.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[GB01] Shafi Goldwasser and Mihir Bellare. *Lecture Notes on Cryptography*. 2001. `http://www-cse.ucsd.edu/~mihir/papers/gb.html`.

[GGvR08] David Galindo, Flavio D. Garcia, and Peter van Rossum. Computational soundness of non-malleable commitments. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *In 4th Information Security Practice and Experience Conference (ISPEC'08)*, volume 4266 of *Lecture Notes in Computer Science*, pages 361–376. Springer Verlag, 2008.

[GH05a] Flavio D. Garcia and Jaap-Henk Hoepman. Off-line karma: A decentralized currency for peer-to-peer and grid applications. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *3th Applied Cryptography and Network Security (ACNS'05)*, volume 3531 of *Lecture Notes in Computer Science*, pages 364–377. Springer Verlag, 2005.

[GH05b] Flavio D. Garcia and Jaap-Henk Hoepman. Off-line karma: A decentralized currency for static peer-to-peer and grid networks. In S. Furnell, P. Dowland, and G. Kormentazas, editors, *5th International Networking Conference (INC'05)*, pages 325–332, Samos Island, Greece, jul 5–7 2005.

[GHPvR05] Flavio D. Garcia, Ichiro Hasuo, Wolter Pieters, and Peter van Rossum. Provable anonymity. In Ralf Küsters and John Mitchell, editors, *3rd ACM Workshop on Formal Methods in Security Engineering (FMSE'05)*, pages 63–72. ACM Press, 2005.

[GM84a] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GM84b] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GMA+95] Steve Glassman, Mark Manasse, Martín Abadi, Paul Gauthier, and Patrick Sobalvarro. The MilliCent protocol for inexpensive electronic commerce. In *Proceedings of the 4th International Conference on the World-Wide-Web*, pages 603–618. O'Reilly, 1995.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th annual ACM conference on Theory of computing (STOC'87)*, pages 218–229. ACM Press, 1987.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *Journal of the ACM*, 38(1):691–729, 1991.

[Gol01] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[GRS96] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: 1st International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 137–150. Springer Verlag, 1996.

[Gut01] Joshua Guttman. Key compromise, strand spaces, and the authentication tests. In *Mathematical Foundations of Programming Semantics 17*, volume 47 of *Electronic Notes in Theoretical Computer Science*, pages 1–21. Elsevier, Amsterdam, 2001.

[GvR06a] Flavio D. Garcia and Peter van Rossum. Completeness of formal hashes in the standard model. Cryptology ePrint Archive, Report 2006/146 (`http://eprint.iacr.org/2006/146`), 2006.

[GvR06b] Flavio D. Garcia and Peter van Rossum. Sound computational interpretation of symbolic hashes in the standard model.

In Hiroshi Yoshiura, Kouichi Sakurai, Kai Rannenberg, Yuko Murayama, and Shinichi Kawamura, editors, *Advances in Information and Computer Security. International Workshop on Security (IWSEC'06)*, volume 4266 of *Lecture Notes in Computer Science*, pages 33–47. Springer Verlag, 2006.

[GvR08] Flavio D. Garcia and Peter van Rossum. Sound and complete computational interpretation of symbolic hashes in the standard model. *Theoretical Computer Science*, 394(1–2):112–133, 2008.

[Her05] Jonathan Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, 340(1):57–81, 2005.

[HO05] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–514, 2005.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of The American Statistical Association*, 58:13–30, 1963.

[Hoe07] Jaap-Henk Hoepman. Distributed double spending prevention. In *Proceedings of the 15th International Workshop on Security Protocols*, 2007.

[JHar] Bart Jacobs and Ichiro Hasuo. Semantics and logic for security protocols. *Journal of Computer Security*, to appear.

[JLM05] Romain Janvier, Yassine Lakhnech, and Laurent Mazar. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer Verlag, 2005.

[Kah67] D. Kahn. *The Codebreakers: The Story of Secret Writing.* Macmillan Publishing Co., 1967.

[KM07] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In Joachim Biskup, editor, *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, Lecture Notes in Computer Science. Springer Verlag, 2007. To appear.

[Low95] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*, pages 147–166. Springer Verlag, 1996.

[Low97] Gavin Lowe. A family of attacks upon authentication protocols. Technical Report 1997/5, Department of Mathematics and Computer Science, University of Leicester, 1997.

[Maz07] Laurent Mazaré. Computationally sound analysis of protocols using bilinear pairings. In Riccardo Focardi, editor, *Preliminary Proceedings of the 7th International Workshop on Issues in the Theory of Security (WITS'07)*, pages 6–21, 2007.

[Mer78] Ralph C. Merkle. Secure communication over insecure channels. *Communications of the ACM*, 21(4):294299, 1978.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In Pierangela Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communication Security (CCS'01)*, pages 245–254. ACM Press, 2001.

[MP05] Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In Joe Kilian, editor, *Proceedings of the 2nd Theory of Cryptography Conference (TCC'05)*, volume 3378 of *Lecture Notes in Computer Science*, pages 169–187. Springer Verlag, 2005.

[MVdV04] Sjouke Mauw, Jan Verschuren, and Erik P. de Vink. A formalization of anonymity and onion routing. In *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS'04)*, volume 3193 of *Lecture Notes in Computer Science*, pages 109–124, 2004.

[MVdV07] Sjouke Mauw, Jan Verschuren, and Erik P. de Vink. Data anonymity in the foo voting scheme. In *Proceedings of the 2nd International Workshop on Views on Designing Complex Architectures (VODCA'06)*, volume 168 of *Electronic Notes in Theoretical Computer Science*, pages 5–28, 2007.

[MW04a] Daniele Micciancio and Bogdan Warinschi. Completeness theorems of the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

[MW04b] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *Proceedings of the 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer Verlag, 2004.

[NLRC98] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the internet – the POPCORN project. In *Proceedings of the The 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 592–601. IEEE, 1998.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attack. In *Procceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM, 1990.

[OO99] K. Ohta and T. Okamoto. Multi-signature scheme secure against active insider attacks. In *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, pages E82–A(1): 21–31, 1999.

[Pár05] Róbert Párhonyi. *Micro Payment Gateways*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, The Netherlands, 2005.

[Pau98] L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, version 0.22, 2000.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM Conference on Computer and Communication Security (CCS'00)*, pages 245–254, 2000.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pages 161–172. ACM Press, 2001.

[Riv04] Ronald L. Rivest. Peppercoin micropayments. In Ari Juels, editor, *Proceedings of the 8th International Conference on Financial Cryptography (FC'04)*, volume 3110 of *Lecture Notes in Computer Science*, pages 2–8. Springer Verlag, 2004.

[RR98] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[RS92] C. Rackoff and D.R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, 1992.

[RS97] Ronald L. Rivest and Adi Shamir. PayWord and MicroMint: Two simple micropayment schemes. In Mark Lomas, editor, *Proceedings of the International Workshop on Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 69–87. Springer Verlag, 1997.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Proceedings of the 11th Fast Software Encryption (FSE'04)*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer Verlag, 2004.

[RSA78a] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RSA78b] R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and publickey cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Ser04] Andrei Serjantov. *On the Anonymity of Anonymity Systems*. PhD thesis, University of Cambridge, 2004.

[Shm04] Vitaly Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3):355–377, 2004.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[SMLN+03] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

[Tsu92] G. Tsudik. Message authentication with oneway hash functions. *ACM Computer Communications Review*, 22(5):29–38, 1992.

[VCS03] Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gun Sirer. KARMA: a secure economic framework for peer-to-peer resource sharing. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*. Papers published on Website: `http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html`, 2003.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 80–91, 1982.

[YGM03] Beverly Yang and Hector Garcia-Molina. PPay: micropayments for peer-to-peer systems. In Vijay Atluri and Peng Liu, editors, *Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS'03)*, pages 300–310. ACM Press, 2003.

[ZHR+04] Ben Y. Zhao, Ling Huang, Sean C. Rhea, Jeremy Stribling, Anthony D Joseph, and John D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

Samenvatting

In onze moderne samenleving worden we omringd door gedistribueerde systemen. De meeste elektronische apparaten die tegenwoordig verkrijgbaar zijn, kunnen in een netwerk functioneren of met elkaar communiceren. Nu is communicatie over een gezamenlijk medium inherent onveilig. Daarom is het goed ontwerpen van veiligheidsprotocollen een prioriteit. Het ontwerpen en analyseren van veiligheidsprotocollen is een uitdagende taak. Diverse protocollen die in de literatuur voorgesteld zijn, bleken later toch gebreken te vertonen. Dit komt door de intrinsieke complexiteit die een kwaadaardige aanvaller met zich meebrengt.

Het traditionele complexiteitstheoretische aanvallersmodel is realistisch, maar ingewikkeld. Hierdoor is het ontwerpen en analyseren van protocollen in dit model gevoelig voor fouten.

Het Dolev-Yao model is een aanvallersmodel waarin de aanvaller de volledige controle heeft over het communicatiemedium. In dit model zijn er geen restricties op de looptijd van de aanvaller, maar is een aanvaller helemaal niet in staat enig cryptografische primitief te breken. Bewijzen in dit model zijn eenvoudiger dan in het complexiteitstheoretische model, maar ondervangen nog steeds de meeste gemaakte fouten in het ontwerpen van veiligheidsprotocollen.

Dit proefschrift bestudeert het probleem van het ontwerpen van veiligheidsprotocollen zowel vanuit het computationele als vanuit het formele gezichtspunt en beschrijft de relatie hiertussen. We bekijken vier originele bijdragen.

- We beschrijven een gedecentraliseerde digitale munteenheid voor peer-to-peer en grid toepassingen dat dubbele uitgaven en andere vormen van fraude kan detecteren.

- We ontwikkelen een formeel raamwerk voor de analyse van anonimizerende protocollen in termen van epistemische logica. We illustreren deze aanpak door zender-anonimiteit en niet-koppelbaarheid voor enkele welbekende protocollen te analyseren.

- We relateren het Dolev-Yao model, uitgebreid met hash functies, met

een realistisch computationeel model. We gebruiken een specifieke gerandomizeerde constructie om hashes te interpreteren. We laten zien dat dit model gezond en volledig is als de aanvaller passief is. We laten zien dat het niet gezond is als de aanvaller actief is.

- We breiden het onderzoek naar deze relatie uit met commitment schema's en actieve aanvallers. We stellen een nieuw, sterker veiligheidsbegrip voor en geven een nieuwe constructie die die veilig bewezen kan worden onder deze definitie. We lichten de bruikbaarheid van deze constructie toe door er een gezond interpretatie van te geven in het standaard model.

Curriculum Vitae

Born on 25 October 1978, Buenos Aires, Argentina.

1992–1996 Joaquín V. Gonzáles High School, La Falda, Córdoba, Argentina.

1997–2002 Computer Science Licentiate (equivalent to MSc.), Faculty of Mathematics, Astronomy and Physics (Fa.M.A.F.). National University of Córdoba (UNC), Argentina.

2003–2007 PhD student at the Security of Systems Group, Radboud University Nijmegen, The Netherlands.

2007– Postdoc at the Digital Security Group, Radboud University Nijmegen, The Netherlands.

Titles in the IPA Dissertation Series since 2002

M.C. van Wezel. *Neural Networks for Intelligent Data Analysis: theoretical and experimental aspects.* Faculty of Mathematics and Natural Sciences, UL. 2002-01

V. Bos and J.J.T. Kleijn. *Formal Specification and Analysis of Industrial Systems.* Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2002-02

T. Kuipers. *Techniques for Understanding Legacy Software Systems.* Faculty of Natural Sciences, Mathematics and Computer Science, UvA. 2002-03

S.P. Luttik. *Choice Quantification in Process Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-04

R.J. Willemen. *School Timetable Construction: Algorithms and Complexity.* Faculty of Mathematics and Computer Science, TU/e. 2002-05

M.I.A. Stoelinga. *Alea Jacta Est: Verification of Probabilistic, Real-time and Parametric Systems.* Faculty of Science, Mathematics and Computer Science, KUN. 2002-06

N. van Vugt. *Models of Molecular Computing.* Faculty of Mathematics and Natural Sciences, UL. 2002-07

A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems.* Faculty of Science, Mathematics and Computer Science, KUN. 2002-08

R. van Stee. *On-line Scheduling and Bin Packing.* Faculty of Mathematics and Natural Sciences, UL. 2002-09

D. Tauritz. *Adaptive Information Filtering: Concepts and Algorithms.* Faculty of Mathematics and Natural Sciences, UL. 2002-10

M.B. van der Zwaag. *Models and Logics for Process Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-11

J.I. den Hartog. *Probabilistic Extensions of Semantical Models.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2002-12

L. Moonen. *Exploring Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2002-13

J.I. van Hemert. *Applying Evolutionary Computation to Constraint Satisfaction and Data Mining.* Faculty of Mathematics and Natural Sciences, UL. 2002-14

S. Andova. *Probabilistic Process Algebra.* Faculty of Mathematics and Computer Science, TU/e. 2002-15

Y.S. Usenko. *Linearization in μCRL.* Faculty of Mathematics and Computer Science, TU/e. 2002-16

J.J.D. Aerts. *Random Redundant Storage for Video on Demand.* Faculty of Mathematics and Computer Science, TU/e. 2003-01

M. de Jonge. *To Reuse or To Be Reused: Techniques for component composition and construction.* Faculty of Natu-

ral Sciences, Mathematics, and Computer Science, UvA. 2003-02

J.M.W. Visser. *Generic Traversal over Typed Source Code Representations.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2003-03

S.M. Bohte. *Spiking Neural Networks.* Faculty of Mathematics and Natural Sciences, UL. 2003-04

T.A.C. Willemse. *Semantics and Verification in Process Algebras with Data and Timing.* Faculty of Mathematics and Computer Science, TU/e. 2003-05

S.V. Nedea. *Analysis and Simulations of Catalytic Reactions.* Faculty of Mathematics and Computer Science, TU/e. 2003-06

M.E.M. Lijding. *Real-time Scheduling of Tertiary Storage.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-07

H.P. Benz. *Casual Multimedia Process Annotation – CoMPAs.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-08

D. Distefano. *On Modelchecking the Dynamics of Object-based Software: a Foundational Approach.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2003-09

M.H. ter Beek. *Team Automata – A Formal Approach to the Modeling of Collaboration Between System Components.* Faculty of Mathematics and Natural Sciences, UL. 2003-10

D.J.P. Leijen. *The λ Abroad – A Functional Approach to Software Components.* Faculty of Mathematics and Computer Science, UU. 2003-11

W.P.A.J. Michiels. *Performance Ratios for the Differencing Method.* Faculty of Mathematics and Computer Science, TU/e. 2004-01

G.I. Jojgov. *Incomplete Proofs and Terms and Their Use in Interactive Theorem Proving.* Faculty of Mathematics and Computer Science, TU/e. 2004-02

P. Frisco. *Theory of Molecular Computing – Splicing and Membrane systems.* Faculty of Mathematics and Natural Sciences, UL. 2004-03

S. Maneth. *Models of Tree Translation.* Faculty of Mathematics and Natural Sciences, UL. 2004-04

Y. Qian. *Data Synchronization and Browsing for Home Environments.* Faculty of Mathematics and Computer Science and Faculty of Industrial Design, TU/e. 2004-05

F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-06

L. Cruz-Filipe. *Constructive Real Analysis: a Type-Theoretical Formalization and Applications.* Faculty of Science, Mathematics and Computer Science, KUN. 2004-07

E.H. Gerding. *Autonomous Agents in Bargaining Games: An Evolutionary Investigation of Fundamentals, Strategies, and Business Applications.* Faculty of Technology Management, TU/e. 2004-08

N. Goga. *Control and Selection Techniques for the Automated Testing of Reactive Systems.* Faculty of Mathematics and Computer Science, TU/e. 2004-09

M. Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs.* Faculty of Science, Mathematics and Computer Science, RU. 2004-10

A. Löh. *Exploring Generic Haskell.* Faculty of Mathematics and Computer Science, UU. 2004-11

I.C.M. Flinsenberg. *Route Planning Algorithms for Car Navigation.* Faculty of Mathematics and Computer Science, TU/e. 2004-12

R.J. Bril. *Real-time Scheduling for Media Processing Using Conditionally Guaranteed Budgets.* Faculty of Mathematics and Computer Science, TU/e. 2004-13

J. Pang. *Formal Verification of Distributed Systems.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-14

F. Alkemade. *Evolutionary Agent-Based Economics.* Faculty of Technology Management, TU/e. 2004-15

E.O. Dijk. *Indoor Ultrasonic Position Estimation Using a Single Base Station.* Faculty of Mathematics and Computer Science, TU/e. 2004-16

S.M. Orzan. *On Distributed Verification and Verified Distribution.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2004-17

M.M. Schrage. *Proxima - A Presentation-oriented Editor for Structured Documents.* Faculty of Mathematics and Computer Science, UU. 2004-18

E. Eskenazi and A. Fyukov. *Quantitative Prediction of Quality Attributes for Component-Based Software Architectures.* Faculty of Mathematics and Computer Science, TU/e. 2004-19

P.J.L. Cuijpers. *Hybrid Process Algebra.* Faculty of Mathematics and Computer Science, TU/e. 2004-20

N.J.M. van den Nieuwelaar. *Supervisory Machine Control by Predictive-Reactive Scheduling.* Faculty of Mechanical Engineering, TU/e. 2004-21

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue.* Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms.* Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications.* Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures.* Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance.*

Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability.* Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions.* Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments.* Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages.* Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics.* Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems.* Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes with Replication.* Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell.* Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model.* Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems.* Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems.* Faculty of Mathematics and

Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality.* Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting.* Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs.* Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations.* Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data.* Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space.* Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices.* Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization.* Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML.* Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models.* Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications.* Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols.* Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition.* Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications.* Faculty of Electrical

Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems.* Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra.* Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation.* Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures.* Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains.* Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use.* Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems.* Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML.* Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery.* Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding.* Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks.* Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data.* Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy.* Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS.* Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems.* Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates.* Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Verification of Optimistic Fair Exchange Protocols.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines.* Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras.* Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit.* Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks.* Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments.* Faculty of Science, Mathematics and Computer Science, RU. 2008-14