UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Formal Methods for the Analysis of Security Protocols

Pedro Miguel dos Santos Alves Madeira Adão
(Licenciado)

Dissertação para a obtenção do Grau de Doutor em Matemática

DOCUMENTO PROVISÓRIO

Junho 2006

UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Formal Methods for the Analysis of Security Protocols

Pedro Miguel dos Santos Alves Madeira Adão
(Licenciado)

Dissertação para a obtenção do Grau de Doutor em Matemática

DOCUMENTO PROVISÓRIO

Junho 2006

Métodos Formais para o Estudo de Protocolos de Segurança

Nome: Pedro Miguel dos Santos Alves Madeira Adão

Doutoramento em: Matemática

Orientador: Professor Doutor Paulo Alexandre Carreira Mateus

Co-Orientador: Professor Doutor Andre Scedrov

Provas Concluídas em:

Resumo: Várias abordagens têm sido propostas nos últimos anos para o estudo de problemas de Criptografia. Visto que o modelo da Criptografia Computacional é bastante complexo e por isso bastante difícil de utilizar, várias abstracções têm sido propostas nos últimos anos sendo a mais bem sucedida a proposta por Dolev-Yao. No entanto, apesar desta abstracção facilitar o tratamento dos problemas, é necessário verificar se estas abstracções são correctas, i.e., se os protocolos cuja correcção foi provada, segundo estas abstracções, são correctos aquando da sua implementação.

Nesta dissertação, começamos por considerar a lógica de indistinguibilidade proposta por Abadi e Rogaway e mostraremos que dois problemas emanam deste resultado: primeiro, não é possível tratar protocolos que tenham ciclos de chaves de encriptação; segundo, a suposição de que o sistema criptográfico consegue ocultar o tamanho da mensagem encriptada é uma suposição muito forte. Nesta dissertação resolvemos ambos os problemas. Para resolver o primeiro enriquecemos o modelo computacional utilizando primitivas criptográficas mais poderosas; para resolver o segundo consideramos uma classe de lógicas mais geral.

A segunda contribuição desta dissertação é a apresentação de uma álgebra de processos semelhante ao calculo pi com comunicação segura, certificados, mas sem uso explícito de criptografia. Nesta linguagem, propriedades de segurança podem facilmente ser estudadas usando equivalência de traços do sistema e equivalência observacional. Apresentamos ainda uma implementação que é simultaneamente correcta e completa face ao modelo da criptografia computacional.

Por último, apresentamos ainda nesta dissertação uma outra álgebra de processos, também similar ao cálculo pi, que nos permite expressar e estudar propriedades de protocolos de segurança quântica.

Palavras Chave: Abstracções Correctas de Criptografia, Álgebras de Processos, Ciclos de Chaves Criptográficas, Criptografia, Segurança Clássica, Segurança Quântica.

Formal Methods for the Analysis of Security Protocols

Abstract: As Computational Cryptography is hard to deal manually, several abstractions have been proposed to analyse security protocols, being one of the most successful the Dolev-Yao abstraction. However, one should investigate how reliable are such abstractions, hence the need to relate these two approaches.

In this dissertation we start by considering the original Abadi-Rogaway logic of formal encryption and its soundness result, observing then that this result has two weaknesses. The first is that it cannot tolerate key-cycles, and the second is that the assumption of length-concealing encryption scheme is too strong. We fix both these problems, the former strengthening the computational model, and the latter by considering a more general class of logics.

The second contribution of this dissertation is the proposal of a language that is variant of the pi-calculus with secure communications, mobile names, and high-level certificates, but with no explicit cryptography. Within this language, security properties can be conveniently studied using trace properties and observational equivalence in the presence of active adversaries. We provide a concrete implementation that is both sound and complete with respect to computational cryptography.

Finally, and arguably one step ahead of reality, we introduce a language also similar to the pi-calculus, that enable us to express and study security properties of quantum cryptographic protocols.

Keywords: Cryptography, Key-Cycles, Process Calculi, Quantum Security, Security, Sound Abstractions of Formal Cryptography.

Acknowledgments

It would be difficult, if not impossible, to mention everyone that contributed and helped me throughout these last four years. To all of them I would like to thank for their contribution. Some of them deserve a special thanks, due to some small "special" things.

Firstly, I would like to thank my supervisors Paulo Mateus and Andre Scedrov for their guidance, support and constant motivation throughout these four years. Their constant shepherding was essential for my development as a researcher. Looking back, I must say that I could not have asked them for more.

Secondly, I would like to thank Cédric Fournet for accepting to supervise me during my internship at Microsoft Research. During this last year I learned more about process calculus and cryptography than I ever imagined to be able to. His guidance, willingness to help, and constant vigilance were the key point for the conclusion of the work in Chapter 3.

Special thanks to Amílcar Sernadas that, without any formal role, carefully overlooked my research steps and constantly suggested new directions for my research.

Special thanks are due to my research fellows Gergei Bana and Jonathan Herzog with whom I had the pleasure of sharing many hours of research, and who were always willing to help a younger PhD student. The results in Chapter 2 are the result, together with Andre Scedrov, of those hours of research.

I would like to thank everyone at the Center for Logic and Computation, CLC, for their fellowship and constant interest in my work. Also to the organisers of the Logic and Computation Seminar Amílcar Sernadas and Carlos Caleiro for inviting me to present my work.

I am also indebted to everyone in the Math Department of UPenn for hosting me for one year.

In these last four years many other people were responsible for keeping my (in)sanity and many are responsible for the completion of this task. I want to thank everyone that made my life in Philadelphia a "once in a lifetime" experience. I would like to thank Beth, Elena, Pilar, Sarah, Sue, Jimmy, Jonathan, Martin, Mike, and specially the restricted Portuguese clan Gabi, Susana, Mario, and the Brazilian infiltrate Daniel ;-).

Finally to all my friends in Portugal that, even when I was abroad, always cared about me. I want to thank Ana, Carla, Cátia, Margarida, Maria José, Patrícia, Rita, Rita, Sara, Teresa, Alexandre, João, João, Luis, Pedro, Pedro, Ricardo, Tiago, and Vitor for their constant support and with whom I shared so many (very) good moments in these last four years.

Finally I would like to thank my family, mom, dad, and my brother Ricardo without whom this was not possible, and to Ana for her presence, love and (infinite) support. Their love was essential for the completion of this (hard) task. All this work is dedicated to them.

The following entities supported my work during the last four years:

- Center for Logic and Computation through research project POCI/MAT/55796/2004 (Quantlog) and POCTI/2001/MAT/37239 (FibLog)
- Fundação para a Ciência e a Tecnologia research grant FCT SFRH/BD/8148/2002

Contents

Chapter 1

Introduction

Cryptography (both classical and quantum), and more generally security, has been a topic of the uttermost interest in the last three decades. Despite the interesting research problems within this field, this increasing interest cannot be separated from the emergence of the Internet, large computer networks, e-commerce and e-government, which demanded the design of better and safer cryptographic protocols.

In the early researches, cryptographic protocols were just devoted to the communication of secret messages [MvOV96, Sti95], but nowadays they are expected to perform several other tasks such as digital signatures [RSA78], message authentication [NS78], secret sharing [Rab81, Sha79], secret key-exchange [Mea92], contract signing [CKS01], electronic cash [Bra99] and electronic voting [FOO92].

When specifying and designing cryptographic protocols, one always supposes that the protocol will run in an adversarial environment and, when trying to analyse the security of such protocols, one may have several different runs at the same time, possibly interleaving with one another. Proving security of such large security protocols easily turns into a task that cannot be performed by hand, hence several tools have been designed to deal with such complex problem [BMV05, MMS97, Pau98, Mea92, Bla01].

The need for such (automated) tools is not only due to the complexity of the protocols. Even small protocols can be difficult to analyse by hand. If we consider the simple protocol proposed by Needham and Schröeder in 1978 [NS78], it was not until 1995 that Lowe [Low95] discovered his famous attack on the protocol and suggested a fix using the CSP-model checker FDR [Low96]. This was the advent of the use of automated tools, model-checkers and theorem provers, for studying security protocols. The proof of security for this (corrected) protocol (called NSL) was done assuming that cryptography is perfect, that is, that one can only decrypt a message if in possession of the correct decryption key. With this "perfect cryptography" assumption, the protocol and the model of the adversary are simple enough to be easily analysed by standard model-checkers and theorem provers. This is one of the most common refinements when analysing security protocols but there are others, e.g., bound the maximum number of instances of the protocol running at the same time.

One question that remained unanswered was the following: What happens when cryptography is not perfect (which is the case in real life where we can always guess a key with small probability)? How secure is NSL when using this “imperfect cryptography”? After some positive results regarding cryptographic soundness of such abstractions, Warinschi [War03] showed that the attack discovered by Lowe against the NS protocol, could also be performed against the corrected version if a somewhat weak, but standard, encryption scheme is used (the El-Gamal encryption scheme [Elg85]). This result turned the attention of the research community for the need of soundness results for formal cryptography, with respect to computational cryptography, i.e., it is important to characterise when protocols proved correct using perfect cryptography are correct when implemented with computational cryptography.

1.1 Background

In the area of cryptographic protocols, two models are noteworthy for their natural definitions and rigorous proofs. The first of these models, the *Computational Model* [GM84, Yao82], is derived from complexity theory. Its definitions are phrased in terms of the asymptotic behaviour of Turing machines, and its main proof technique is reduction. The second model is the *Symbolic Model (or, Formal Model, or Dolev-Yao model)* [DY83, NS78], is so-named because of its genesis in the field of formal methods. Its definitions are phrased in terms of process algebras and state machines (particularly non-deterministic ones) and it uses many different proof methods (including automated ones). There are many differences between the two models , but two in particular are key: their representations of messages and the power they give to the adversary.

- In the computational model, messages are families of probability distributions over bitstrings (indexed by the security parameter). The adversary is modelled as an algorithm of realistic computational power: probabilistic polynomial-time, PPT.
- The formal model imposes a more complex structure. Messages are expressions built according to a particular grammar. The atomic messages are symbols representing keys, random values, texts, and so on. More complex messages can be built from simpler ones by application of (symbolic) functions, e.g., pairing and encryption. The adversary is given only limited power to manipulate these expressions, such as separating a concatenation or decrypting an encryption (if it knows the decrypting key). These possible operations are specified via a set of equations.

While the former is a more concrete approach (based on complexity theory) with limits in the computational power of the adversary, concrete cryptographic algorithms that work on bitstrings, computational indistinguishability as the “equivalence notion”, and precise (and accepted) definitions of security goals (which provides a common ground for discussion), the latter requires a higher-level of abstraction but in exchange provides automatic tools, and is easier to handle. Examples of such tools range from logics [AT91, Syv91, SM93, GM95, BAN96, SC00, IK03, CVB04, CVB05, DDM$^+$05, CMP05], to theorem provers [Pau97b, Pau97a, Pau98, Bla06] and model checkers [Low96, MMS97, Bla01, BMV05, BAF05], process algebras [AG99, AG97, AG98], and strand spaces [FHG98, FHG99]. The major drawback of such abstractions is that, as seen above, it is possible to prove the correctness of protocols that are susceptible to attacks in a concrete implementation.

Despite these differences, certain intuitions can be translated between the two models in the expected way. In particular, under carefully chosen conditions, *indistinguishability of messages* can be mapped directly from one model to the other.

1.1.1 The Abadi-Rogaway Logics of Formal Encryption

Relation between these two models was first demonstrated by Abadi and Rogaway [AR00, AR02] in a particular setting and under strong assumptions. In their formulation of the formal model (where messages are constructed from basic terms, blocks and keys, via pairing and encryption) two expressions are said to be indistinguishable to the adversary (also called *formally equivalent*) if their only differences lie in the encryption terms that cannot be decrypted by the formal adversary. In the computational model, on the other hand, messages are families of probability distributions on bit-strings. Equivalence of computational messages is captured by the standard notion of computational indistinguishability (i.e., indistinguishability by an efficient algorithm [GM84]).

Relating the two models.

Once a computational encryption scheme is fixed, an intuitive function translates expressions between the two models. This function (called *interpretation*) maps blocks to fixed bitstrings, keys to bitstrings generated by the key-generation algorithm, pairs to the pairing of the interpretations, and encryptions to the bitstring that result from running the encryption algorithm on the interpretation of the encrypted message.

This interpretation maps each formal expression to an ensemble (indexed by the security parameter) of probability distributions over bit-strings. Given an encryption scheme, and therefore a particular interpretation function, one can then ask whether all pairs of equivalent formal messages map to indistinguishable ensembles of probability distributions. If so, it is said that *soundness* holds and it implies that the formal model is a faithful abstraction of the computational model: security in the formal model implies security in the computational model as well. If the converse holds, that is, if every pair of indistinguishable probability distributions corresponds to interpretations of equivalent formal messages, we say that *completeness* holds. In this case, we have that the formal model is not over conservative, that is, it encompasses only the subtleties of the computational model.

In their seminal work, Abadi and Rogaway demonstrated (in the symmetric-key encryption setting) that soundness holds when the security level of the computational encryption algorithm is ‘type-0’ (a scheme is type-0 if it does not leak any information about the size of the encrypted plaintext, and given two ciphertexts one cannot say if they correspond to the encryption of the same message, nor if they correspond to the encryption of messages using the same key). This result was later translated to the public-key setting by Micciancio and Warinschi [MW04b], who found that soundness is guaranteed by encryption schemes that satisfy ‘chosen-ciphertext security’ [RS91, Sah99] (CCA-2 in the notation of [BDPR98]). The power of chosen-ciphertext security has been confirmed by subsequent extensions [Her04, CH06]. These results, however—in both the symmetric and asymmetric settings—share two significant weaknesses.

Weaknesses of previous soundness results.

Firstly, the result of Abadi and Rogaway fails to hold in the presence of *key cycles*. An expression has a (symmetric) key cycle if one can find symmetric keys K_1, $K_2 \ldots K_n$ such that K_i is encrypted in the expression under K_{i+1} and K_n is encrypted by K_1. (In the asymmetric setting, the public key K_i encrypts the private key K_{i+1}^{-1}, and K_1 encrypts K_n^{-1}.) The formal model makes no distinction between those messages that have key-cycles and those that do not. Further, the interpretation function is well-defined over key-cycles, and so, formal key-cycles are computationally meaningful. However, neither the soundness result of Abadi and Rogaway nor subsequent soundness results (described in Section 2.6) are known to hold for such messages. (In fact, the stronger of these results [BPW03, CH06] assumes that no private or symmetric keys are encrypted at all!)

Thus, the question of key-cycles is both interesting in its own right and has implications in a larger context. The standard security definitions for computational encryption, such as CCA-2 security, do not obviously imply security in the presence of key-cycles [MRS98]. The formal model, on the other hand, assumes that key-cycles do not weaken encryption in any way. Therefore, the issue of key-cycles may represent an actual 'gap' between the formal and computational models, and thus may shed light on their general relationship.

The majority of the results relating the two models show the formal model to be sound with respect to standard definitions of the computational model—with some notable exceptions. Some gaps have been positively identified. (For example, Canetti and Herzog [CH06] and Backes and Pfitzmann [BP05] have demonstrated that the formal definition of secrecy is strictly weaker than the computational definition.) However, these gaps were 'closed' by forcing changes onto the formal model. Should the resolution of the problem of key-cycles again cause changes to the formal model, or could it this time be more naturally resolved through modifications to the computational model?

The second weakness of the original Abadi-Rogaway result also identifies a possible gap between the two models, but one that has already been previously studied. In particular, the original soundness results of Abadi and Rogaway assumes that formal encryption conceals all aspects of the plaintext. That is, their result requires that symmetric encryption hides (among other things) the length of the plaintext. Unfortunately, this cannot be achieved for many contexts, and this can thus be considered a 'gap' between the two models. This particular gap has already been considered by Micciancio and Warinschi [MW04b], Laud [Lau04], and Micciancio and Panjwani [MP05] who resolve the gap by weakening the formal model. These results, however, are highly specific to particular classes of computational encryption schemes. It is unclear if or how these results can be generalised to consider other encryption schemes that might leak other kinds of information. Rephrasing, can every encryption scheme provide soundness to *some* weakened version of the formal model, or do some 'gaps' remain?

1.1.2 Process Algebras for Security

Process Algebras were introduced as simple models to deal with concurrent systems [Mil89, Hoa80]. In these models, computation is defined as the communication/interaction between processes. The tradition on using process calculus to reason about properties of systems led easily to its use for studying cryptographic protocols.

Pi calculus [MPW92] was used since its early stages to model mobile processes and dynamic channels. Pi calculus' channels are very simple but at the same time a very powerful tool as they can be created and passed among principals. In particular, it is possible to model private and secure networks just using private channels. The extrusion rules of the pi-calculus ensure that as long as a channel is not given to the adversary, he will never be able to access it.

But how can one implement those channels? Private channels are usually implemented using cryptography and this cannot be expressed in the pi calculus. A first approach to include cryptographic primitives in a process algebra was the development of spi calculus [AG99, AG97, AG98]. Spi calculus is an extension of pi calculus that has in-built cryptographic primitives. This extension allows the explicit representation of cryptography in protocols. With that, we can express security properties as equivalences, e.g., secrecy of V is shown by proving that a process that sends V is equivalent to one that sends V'. Modelling an adversary is also very easy in this setting: we just allow any context to be an adversary. By doing this, we do not need to specify the capabilities of an adversary; we just allow him to do everything. Two protocols are then said to be equivalent if their observational behaviour is equivalent. This trend was very successful. The caveat is that, in spite of the strong (symbolic) guarantees provided by such methods, no cryptographic guarantees are given for the protocols proved correct. Real implementations use probabilistic encryption and guessing a key is something that is always possible in real life, but excluded in the spi calculus.

Another extension is the applied pi calculus [AF01]. The applied pi is another extension of the pi calculus that includes not only cryptographic primitives but arbitrary operations and equations.

Type systems for process calculus have also been applied to the study of security [Aba99, HVY00, BDNN01, GJ03, GJ04, BBD$^+$05, CGG05, AB05, Lau05]. This method allows a static-analysis of infinite-state protocols providing an alternative to finite-state model-checkers.

Another interesting approach is to supplement process calculi with concrete probabilistic or polynomial-time semantics. Unavoidably, reasoning on processes becomes more difficult. For example, Lincoln, Mitchell, Mitchell, and Scedrov [LMMS98] introduce a probabilistic process algebra for analysing security protocols, such that parallel contexts coincide with probabilistic polynomial-time adversaries.

In this framework, further extended by Mitchell, Ramanathan, Scedrov, and Teague [MRST01, MRST04, MRST06], they develop an equational theory and bisimulation-based proof techniques. A general simulatability theorem is presented by Mateus, Mitchell and Scedrov [MMS03]. An automated tool for this approach has been proposed recently by Blanchet [Bla06]. By application of games, automatically or with assistance from the user, one is able to prove correctness of protocols specified in an extension of this calculus.

1.1.3 Quantum Security

As for quantum cryptography two seminal works have driven most of the research on this area: the quantum polynomial time factorisation algorithm proposed by Shor [Sho97]; and the quantum public key agreement protocol BB84, proposed by Bennett and Brassard [BB84], which was proved to be perfectly secure by Shor and Preskill [SP00]. While Shor's algorithm raises the threat of making widely used cryptographic systems completely obsolete by a breakthrough in quantum hardware, the BB84 protocol shows that quantum communication channels allow public perfect security.

The fact that we are still far from an implementation of quantum computers (at the present stage, quantum computers can only work with a few qubits) does not make the field of quantum security useless. In fact, several commercial application of quantum cryptographic devices already exist. For instance, there already exist encryption devices that combine Quantum Key Distribution algorithms with AES (Advanced Encryption Standard) to achieve perfect security in Shannon's sense, and Quantum Random Number Generators that can be included in any standard computer through a PCI card.

Being able to analyse quantum cryptographic protocols that include complex interactions of different cryptographic primitives, and more generally quantum programs, is a step that needs to be taken in the near future either by adapting current techniques from classical cryptography or by creating new methods that intrinsically incorporate the quantum phenomena.

1.2 Our Work

This dissertation has three different contributions.

1.2.1 Bridging the Gap Between Formal and Computational Cryptography

In Chapter 2, we address the two weaknesses of the original Abadi-Rogaway result mentioned above. First, we consider the problem of key-cycles and show that an actual gap exists, but one that can be bridged by strengthening the computational model. (We note that this is the first gap to be closed in this way, rather than by weakening the formal model.) In doing this, however, we must assume (unlike Abadi and Rogaway) that formal encryptions reveal two things: the 'length' of their plaintexts, and whether two different ciphertexts were created using the same key. With this as motivation, we then turn to generalisations of the Abadi-Rogaway formalism. In particular, we show (in a general way) how Abadi and Rogaway's formulation of the formal model can be expanded to consider encryption schemes (computational or information theoretic) that leak partial information such as plaintext-length. That is, we investigate the conditions under which a computational encryption scheme provides soundness and completeness to a (possibly weakened) version of the formal model.

The Problem of Key Cycles:

We solve the issue of soundness in the presence of key-cycles by using the notion of *key-dependent message* (KDM) security for symmetric encryption. This definition was recently introduced simultaneously by Black, Rogaway and Shrimpton [BRS02], who consider it in their own right, and by Camenisch and Lysyanskaya [CL01], who use it for an implementation of a credential system that discourages people from transferring credentials. We will, however, use it to demonstrate two points:

1. Firstly, as predicted by Black *et al.*, we show that this new notion of KDM security is strong enough to achieve soundness in the presence of key cycles (for a somehow weaker version of the original model proposed by Abadi and Rogaway);

2. Secondly, we show that in order to achieve soundness for formal encryption, we need stronger computational definitions than the ones used by Abadi and Rogaway. In particular, we show that both soundness and KDM security neither imply nor are implied by type-0 security.
   (In Appendix B we address the problem of key-cycles in the case of asymmetric encryption and show that also in the case of asymmetric encryption, chosen-ciphertext (CCA-2) security, which is the strongest known definition of security in the (standard) computational model, neither implies nor is implied by soundness or (asymmetric) KDM security.)

Thus, the problem of key-cycles was, in fact, a genuine gap between the formal and computational models at the time of the original Abadi-Rogaway result, but one that can be repaired using recent advances in the computational model. (We believe this to be the first time that a gap has been bridged by modifying the computational model rather than the formal one.)

Unfortunately, our results regarding key-cycles serve also to demonstrate another gap between the formal and computational models—one that must also be closed by weakening the formal model. In particular, KDM-security allows a ciphertext to reveal two things: the bit-length of the plaintext, and the identity (but not value) of the key used in the encryption. Therefore, soundness for key-cycles requires that encryptions in the formal model also reveal these two things.

This fact leads to the other weakness of the original Abadi-Rogaway result: it assumes that computational encryption can hide *all* aspects of the plaintext. In particular, it demonstrates that soundness is provided by ‘type-0’ encryption, which hides (among other things) the length of the plaintext. However, most encryption schemes do not hide this fact, and it can be argued that ‘type-0’ encryption is impossible in general. For this reason, the original Abadi-Rogaway result must be generalised to consider the kinds of soundness that can be provided by real encryption schemes.

The Problem of Leakage of Partial Information:

More specifically, we extend the applicability of the Abadi-Rogaway treatment by expanding their formulation of the formal model. We show how to adjust the formal notion of equivalence in order to maintain soundness when the underlying computational encryption scheme leaks partial information. Furthermore, we investigate the circumstances under which an encryption scheme (or security definition) can be thought of as implementing *a* (possibly weakened) version of the formal model.

Also, our treatment will capture both the standard complexity-based encryption schemes of the computational model and probabilistic, *information-theoretic* encryption schemes. That is, we use a general probabilistic framework that includes, as special cases, both the computational and purely probabilistic encryption schemes (such as One-Time Pad).

We consider not only soundness properties, but also we provide *completeness* theorems. In this context, an encryption scheme provides soundness if, when used in the interpretation function, equivalent formal messages become indistinguishable probability distributions. On the other hand, a scheme provides completeness if whenever two formal messages have indistinguishable interpretations, they are equivalent. Our generalisation will show how both of these conditions can be maintained. Key-cycles do not pose a problem for completeness, hence we will only discuss completeness in relation to leaking of information.

1.2.2 Cryptographically Sound Implementation for Communicating Processes

In Chapter 3, we develop a first sound and complete implementation of a distributed process calculus (We refer the reader to [AF06a] for the discussion related to soundness). Our calculus is a variant of the pi calculus; it provides name mobility, reliable messaging and authentication primitives, but neither explicit cryptography nor probabilistic behaviours. Taking advantage of concurrency theory, it supports simple reasoning, based on labelled transitions and observational equivalence. We precisely define its concrete implementation in a computational setting. We establish general soundness and completeness results in the presence of active adversaries, for both trace properties and observational equivalences, essentially showing that high level reasoning accounts for all low-level adversaries. We illustrate our approach by coding security protocols and establishing their computational correctness by simple formal reasoning.

We implement high-level functionalities using cryptography, not high-level views of cryptographic primitives. Following recent related works, we could instead have proceeded in two steps, by first compiling high-level communications to an intermediate calculus with ideal, explicit cryptography (in the spirit of [AFG02, AFG00]), then establishing the computational soundness of this calculus with regards to computational cryptography. However, this second step is considerably more delicate than our present goal, inasmuch as one must provide a sound implementation for an arbitrary usage of ideal cryptography. In contrast, for instance, our language keeps all keys implicit, so no high-level program may ever leak a key or create an encryption cycle. (We considered targeting existing idealised cryptographic frameworks with soundness theorems, but their reuse turned out to be more complex than a direct implementation.)

Our concrete implementation relies on standard cryptographic primitives, computational security definitions, and networking assumptions. It also combines typical distributed implementation mechanisms (abstract machines, marshaling and unmarshaling, multiplexing, and basic communications protocol.) This puts interesting design constraints on our high-level semantics, as we need to faithfully reflect their properties and, at the same time, be as abstract as possible. In particular, our high-level environments should be given precisely the same capabilities as low-level probabilistic polynomial-time (PPT) adversaries. For example, our language supports abstract reliable messaging: message senders and receivers are authenticated, message content is protected, and messages are delivered at most once. On the other hand, under the conservative assumption that the adversary controls the network, we cannot guarantee message delivery, nor implement private channels (such that some communications may be undetected). Hence, the simple rule $\overline{c}\langle M\rangle.P \mid c(x).Q \rightarrow P \mid Q\{M/x\}$, which models silent communication "in the ether" for the pi calculus, is too abstract for our purposes. (For instance, if P and Q are implemented on different machines connected by a public network, and even if c is a restricted channel, the adversary can simply block all communications.) Instead, we design high-level rules for communications between explicit principals, mediated by an adversary, with abstract labels that enable the environment to perform traffic analysis but not forge messages or observe their payload. Similarly, process calculi feature non-deterministic infinite computations, and we need to curb these features to meet our low-level complexity requirements.

1.2.3 A Process Algebra for Reasoning About Quantum Security

In Chapter 4 we present a process algebra for specifying and reasoning about quantum security protocols. Since the computational power of the protocol agents must be restricted to quantum polynomial-time, we introduce the logarithmic cost quantum random access machine (QRAM) similar to [CR73, Kni96], and incorporate it in the syntax of the algebra. Probabilistic transition systems give the semantic for the process algebra. Term reduction is stochastic because quantum computation is probabilistic and, moreover, we consider a uniform scheduler to resolve non-deterministic choices. With the purpose of defining security properties, we introduce observational equivalence and quantum computational indistinguishability, and show that the latter is a congruence relation. A simple corollary of this result asserts that any security property defined via emulation is compositional. Finally, we illustrate our approach by establishing the concept of quantum zero-knowledge protocol.

The computational model we adopt to define quantum polynomial terms is based on the logarithmic cost random access machine [CR73]. A hybrid model, using both classic and quantum memory, similar to [Kni96] but with complexity assumptions, is considered and it is shown to be (polynomial-time) equivalent to a uniform family of quantum circuits (which are, by themselves, equivalent to quantum Turing machines). Such machines model the computation of each agent, receive qubits as input, and return qubits as output.

Thanks to the non-cloning theorem, quantum information cannot be copied without prior knowledge of its state. This observation imposes some design options in the process algebra, since it is necessary to know which agent possesses a qubit in order to know who can retrieve each piece of information. In order to deal with this fact, a set of agents is fixed and the qubits are partitioned among them.

Although several other approaches to quantum process algebras are already present in the literature (see [GN05], for instance), ours is quite original, due to the universe of application—security protocols. In our approach, process terms are divided into local and global. An agent is modelled by a local process while a protocol is modelled by a global process so, a global process corresponds to local processes running in parallel. A semantics based on probabilistic transition systems (which can be easily translated to Markov chains) is provided, and the probabilistic transitions are defined using rules and assuming a uniform scheduler to resolve non-deterministic choices.

Agent observation is defined as a probability distribution over binary words obtained by measuring on the computational basis (some of) the agent's qubits. This measurement is done at the end of a protocol run. This concept is the key ingredient to establish observational equivalence that, in the context of security protocols, is based on computational indistinguishability [Yao82]. Intuitively, two process terms are observational equivalent for an agent if, after making all possible reductions to each process, it is impossible to distinguish (in quantum polynomial-time) the qubits of the agent on both processes. Since we internalise quantum polynomial-time machines in the process algebra language, observational equivalence is easily defined and it is shown to be a congruence relation.

One of the most successful ways for defining secure concurrent cryptographic tasks is via process emulation [AG99, Can00]. This definitional job boils down to the following: a process realises a cryptographic task if and only if it emulates an ideal process that is known to realise such task. Based on the notion of observational equivalence, we establish the notion of emulation for the quantum process calculus and show that it is compositional. Finally, we provide the notion of quantum zero-knowledge via process emulation.

1.3 Outline of this Dissertation

This dissertation is organised in 4 more chapters and an appendix. It follows a brief outline of each of them.

1.3.1 Soundness of Formal Cryptography

In Chapter 2 we extend the Abadi-Rogaway Logics of Formal Encryption in order to address the problems of key-cycles and leakage of partial information. In Section 2.1, we start by recalling the original result of Abadi and Rogaway [AR02], and the original logic, that is, the definition of the language and definition of the formal equivalence relation. We proceed then with the precise definition of the computational model, in particular, we present the security notion used by Abadi and Rogaway in their result, type-0 security, and afterwards we define the interpretation of a message.

After this brief introduction we start addressing the problem of (symmetric) key-cycles, Section 2.2. We start by showing that type-0 is not strong enough to achieve soundness. We present then the notion of security introduced by Black, Rogaway and Shrimpton [BRS02] that will solve the problems of key cycles (KDM-Security). Before showing that, we have to extend the language of [AR02] and define a new notion of equivalence that is finer than the one in the original result. We prove then that KDM-Security provides soundness for this new language. We conclude by showing that type-0 security does not imply, nor is implied by KDM security.

We proceed then to the analysis of encryption schemes that reveal partial information, Section 2.3. We consider 3 different types of encryption schemes (type-1, encryption schemes that do not conceal the length of the plaintexts, type-2, encryption schemes that do not conceal the fact that two ciphertexts were encrypted with the same key, and type-3, encryption schemes that do not conceal the length of the plaintexts neither the fact that two ciphertexts were encrypted with the same key) and for each of them present the changes that we have to perform in the formal model in order to obtain both soundness and completeness for these examples (without key cycles, in spite of the KDM security notion could be adapted for these cases). In Section 2.4, we consider an implementation of One-Time Pad and show both soundness and completeness results for such implementation.

In Section 2.5, we state our general soundness and completeness theorems for logics of formal encryption. We show that for any logic such that there exists a notion of equivalent non-decryptable terms (called *properness*), soundness holds for any encryption scheme that cannot distinguish the concrete implementations of the terms of each class. We also show that completeness holds, as long as the scheme may distinguish elements of any two different classes. We conclude this section by showing that the examples in Section 2.3 are simple corollaries of these general theorems.

We conclude this Chapter by referring some related work, Section 2.6, and presenting some conclusions and pointers to future work 2.7.

Chapter 2 extends [Ban04, ABS05, ABHS05] and is the result of a collaboration with Gergei Bana, Jonathan Herzog, and Andre Scedrov. Part of this work was done while the author was a visiting student at the University of Pennsylvania.

1.3.2 Cryptographically Sound Implementation of Communicating Processes

In Chapter 3 we present a language, similar to the pi-calculus, with secure communications, mobile names and high-level certificates (but no high-level cryptography), that has a cryptographically sound implementation, that is, security properties can be studied conveniently using trace properties and observational equivalence, and those properties may be carried to a concrete implementation.

We start by describing the low-level target model, Section 3.1, as the constraints imposed by this will drive the design of the high-level language. We then present our high-level language and semantics, Section 3.2. Section 3.3 defines and illustrates our notion of high-level equivalence, in particular, we show how to encode strong secrecy and authentication properties in our language, and how is equivalence related with certificates.

Section 3.4 develops applications. We show how to model anonymous forwarders and exhibit an example of an electronic payment protocol. These examples explore the fact of our language have in-built authentication and signing primitives. In this section, we also show that given any system S, possibly with certificates and names shared among principals, we can always find an initial system S° where principals share no information, such that there is a transition from S° to S. This allow us to consider programs where information is already shared without the need to refer always to initial states.

In Section 3.5 we describe our concrete implementation. This implementation relies on standard cryptographic primitives, computational security definitions, and networking assumptions.

It also combines typical distributed implementation mechanisms (abstract machines, marshaling and unmarshaling, multiplexing, and basic communications protocol.) In Section 3.6 we state our completeness theorems.

We conclude this Chapter by discussing related work, Section 3.7, and presenting our conclusions and possible extensions to our framework 3.8.

Chapter 3 extends [AF06b, AF06a] and is the result of a collaboration with Cédric Fournet. Part of this work was done while the author was a research intern at Microsoft Research Cambridge.

1.3.3 An Extension to Quantum Security

In Chapter 4 we present an algebra for specifying and reasoning about quantum security protocols. In order to restrict the power of the agents to quantum polynomial-time, we include in the syntax of our algebra the logarithmic cost quantum random access machine (QRAM).

We start by describing our process algebra, Section 4.1. In particular, we introduce our notion of Quantum Polynomial Machines, its execution model. Then, we introduce it in the language of out process algebra, and define the semantics for such algebra, as well as the notion of equivalence of processes.

In Section 4.2 we present our emulation theorem . This theorem is immediately derived from the one in [MMS03]. Since quantum computational indistinguishability is a congruence relation, we have that all the properties defined via emulation are compositional. We then illustrate its usage in Section 4.3 by defining the notion of Quantum Zero-Knowledge protocols via emulation.

We conclude this Chapter by discussing related work, Section 3.7, and conclusions of our work 3.8.

Chapter 4 extends [AM] and is the result of a collaboration with Paulo Mateus.

1.3.4 Conclusions

Is Chapter 5 we revise the contributions of this dissertation. We briefly discuss the contributions of each chapter and point out future research directions for each of them.

1.3.5 Appendixes

In this work we include three Appendixes. In the first, Appendix A, we present the main cryptographic definitions used throughout this dissertation. We also prove some properties that, in spite of being used in particular lemmas, are interesting enough to be proven separately.

In Appendix B we consider the problem of key-cycles in the case of asymmetric encryption. This is similar to the case for symmetric encryption and curiously was obtained prior to the former [ABHS05]. The results for symmetric encryption presented in Chapter 2 are a recast of these.

In Appendix C we present the proofs of our completeness results of Chapter 3.

1.4 Claim of Contributions

The contributions of this dissertation are divided in three different topics. We would like to stress those that we think are the main contributions in each topic. As for the relation between formal and computational cryptography our main contributions are:

- solution to the problem of key-cycles that was for a long conjectured to be a gap between the two models; we not only solved the problem but also showed that new cryptographic primitives were needed in order to bridge this apparent gap between the models; as a consequence of this result, since most of the extensions of the Abadi-Rogaway original result use it as a “black-box”, we also remove the referred restriction from this extensions;
- showed that cycles of length 1 are sufficient to differ KDM from CCA2. The question about longer cycles remained an open question until recently [BPS];
- found new relations among different notions of security of encryption schemes, in particular, how does KDM relates with the standard security notions;
- extension of the applicability of the Abadi-Rogaway result to weaker encryption schemes; we showed that, for symmetric encryption, and in the presence of passive adversaries, subtleties of the encryption schemes may be faithfully captured by the formal model;
- we provide a unified framework for both computational and information-theoretic encryption schemes.

As for the cryptographically sound implementations of communicating processes our main contributions are:

- a simple calculus for secure distributed communication with two forms of authentication, expressive enough to program a large class of protocols;
- simple reasoning for this language based on labelled transition systems and observational equivalence;
- concrete implementation for such calculus as a collection of PPT Turing machines that rely on standard cryptographic algorithms, and traditional distributed implementations mechanisms;
- the first cryptographic soundness and completeness results for a distributed process calculus.

As for the study of quantum security, our main contributions are:

- process algebra for specifying and reasoning about quantum security protocols;
- introduction a hybrid model, that uses both classic and quantum memory, with complexity assumptions, that is (polynomial-time) equivalent to a uniform family of quantum circuits;

- defined the notion of computational indistinguishability for the process algebra and showed that it was a congruence relation. As a corollary security properties defined via emulation are compositional.

Chapter 2

Soundness of Formal Encryption

Relating Symbolic and Computational cryptography has attracted the interest of the research community in the last few decades. Several different directions have been taken to bridge the gap between the two models: some extend the existing results by including more primitives; some by adapting existing results from the passive adversary scenario to the active adversary scenario; some others by including new primitives from computational cryptography.

This chapter is one more effort to bridge the gap between these two communities. Mainly, we try to bridge two gaps that exist since the early results of Abadi and Rogaway. The first is the non-existence of soundness results in the presence of key-cycles. Key-cycles do not present a problem from the symbolic point of view. One may even argue that protocols that create messages with encryption cycles may be avoided and are just result of bad engineering. But, even if we restrict our protocols to the cases where no cycles are created, no one can ensure us that an adversary is not able to create cyclic encryptions and that these would not cause problems. Studying this is part of the work in this chapter. We show that it is possible to close this gap but for that we need to use new definitions of security.

The second gap that we try to close is to extend the original Abadi and Rogaway result when the encryption scheme used provides less security guarantees. The encryption scheme used in their original result is very strong and arguably impossible to realise in many contexts. We want then to relax such conditions by allowing the use of weaker encryption schemes but still achieving similar soundness results. We want for instance, to allow encryption schemes that reveal the length of the encrypted plaintext. We study this particular example and then create a uniform framework with which we are able to characterise a large family of encryption schemes.

This chapter is organised as follows: in Section 2.1, we start by recalling the original result of Abadi and Rogaway [AR02]. In Section 2.2 we address the problem of (symmetric) key-cycles. In particular we show that type-0 is not strong enough to achieve soundness, and present the notion of security introduced by Black, Rogaway and Shrimpton [BRS02] that will solve the problems of key cycles (KDM-Security). In Section 2.3 we proceed with the analysis of encryption schemes that reveal partial information. In Section 2.4, we consider an implementation of One-Time Pad and show both soundness and completeness results for such implementation. Section 2.5 is devoted to our general soundness and completeness theorems for logics of formal encryption. As a corollary of these results, we have the examples of Section 2.3. We conclude this Chapter by referring some related work, Section 2.6, and presenting some conclusions and pointers to future work 2.7.

As an extension of Section 2.2, we present in Appendix B similar results for the case of key-cycles in the case of asymmetric encryption.

2.1 The Abadi-Rogaway Soundness Theorem

In this section, we provide the context and background for this chapter. We briefly summarise the main definitions and results of Abadi and Rogaway's original work [AR00, AR02]. In particular, we start presenting the formal model, then describe the computational model, and then introduce the notion of soundness. Furthermore, we also introduce the notion of completeness, which can be viewed as the counter-point to soundness.

2.1.1 The Formal Model

In this model, messages (or *expressions*) are defined at a very high level of abstraction. The simplest expressions are symbols for atomic keys and bit-strings. More complex expressions are created from simpler ones via encryption and concatenation, which are defined as abstract, 'black-box' constructors.

Definition 2.1 (Symmetric Expressions). Let **Keys** $= \{K_1, K_2, K_3, ...\}$ be an infinite discrete set of symbols, called the set of symmetric keys. Let **Blocks** be a finite subset of $\{0,1\}^*$. We define the *set of expressions*, **Exp**, by the grammar:

$$\mathbf{Exp} ::= \mathbf{Keys} \quad | \quad \mathbf{Blocks} \quad | \quad (\mathbf{Exp}, \mathbf{Exp}) \quad | \quad \{\mathbf{Exp}\}_{\mathbf{Keys}}$$

Let **Enc** $::= \{\mathbf{Exp}\}_{\mathbf{Keys}}$. We will denote by *Keys*(M) the set of all keys occurring in M. Expressions of the form $\{M\}_K$ are called *encryption terms*.

Expressions may represent either a single message sent during an execution of the protocol, or the entire knowledge available to the adversary. In this second case, the expression contains not only the messages sent so far, but also any additional knowledge in the adversary's possession.

We wish to define when two formal expressions are indistinguishable to the adversary. Intuitively, this occurs when the only differences between the two messages lie within encryption terms that the adversary cannot decrypt. In order to rigorously define this notion, we first need to formalise when an encryption term is 'undecryptable' by the adversary, which in turn requires us to define the set of keys that the adversary can learn from an expression.

An expression might contain keys in the clear. The adversary will learn these keys, and can then use them to decrypt encryption terms of the expression—which might reveal yet more keys. By repeating this process, the adversary can learn the set of *recoverable decryption keys*:

Definition 2.2 (Subexpressions, Visible Subexpressions, Recoverable Keys, Undecryptable Terms, B-Keys). We define the *set of subexpressions* of an expression M, $sub\,(M)$, as the smallest subset of expressions containing M such that:

- $(M_1, M_2) \in sub(M) \implies M_1 \in sub(M)$ and $M_2 \in sub(M)$, and
- $\{M'\}_K \in sub(M) \implies M' \in sub(M)$.

We say that N is a subexpression of M, and denote it by $N \sqsubseteq M$, if $N \in sub(M)$.

The set of *visible subexpressions* of a symmetric expression M, $vis(M)$, is the smallest subset of expressions containing M such that:

- $(M_1, M_2) \in vis(M) \implies M_1 \in vis(M)$ and $M_2 \in vis(M)$, and
- $\{M'\}_K$ and $K \in vis(M) \implies M' \in vis(M)$.

The *recoverable keys of* a (symmetric) expression M, *R-Keys*(M), are those that an adversary can recover by looking at an expression. That is, *R-Keys*$(M) = vis(M) \cap$ *Keys*(M).

We say that an encryption term $\{M'\}_K \in vis(M)$ is *undecryptable* in M if $K \notin$ *R-Keys*(M). Among the non-recoverable keys of an expression M, there is an important subset denoted by *B-Keys*(M). The set *B-Keys*(M) contains those keys which encrypt the outermost undecryptable terms. Formally, for an expression M, we define *B-Keys*(M) as

$$\textit{B-Keys}(M) = \{K \in \textit{Keys}(M) \mid \{M\}_K \in vis(M) \text{ but } K \notin \textit{R-Keys}(M)\}.$$

Example 2.1. Let M be the following expression

$$((\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}), ((K_2, \{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}), \{K_5\}_{K_2})).$$

In this case, *Keys*$(M) = \{K_1, K_2, K_3, K_4, K_5, K_6, K_7\}$. The set of recoverable keys of M is *R-Keys*$(M) = \{K_2, K_5, K_6\}$, because an adversary sees the non-encrypted K_2, and with that he can decrypt $\{K_5\}_{K_2}$, hence recovering K_5; then, decrypting twice with K_5, K_6 can be revealed. We also have that *B-Keys*$(M) = \{K_3, K_4\}$.

The formal model allows expressions to contain *key cycles*:

Definition 2.3 (Key-Cycles). An expression M contains a *key-cycle* if it contains encryption terms $\{M_1\}_{K_1}, \{M_2\}_{K_2}, \ldots, \{M_n\}_{K_n}$ (where $\{M_i\}_{K_i}$ denotes the encryption of the message M_i with the key K_i) and $K_{i+1} \sqsubseteq M_i$ and $K_1 \sqsubseteq M_n$. In this case we say that we have a key-cycle of length n.

According to our definition, expressions such as $\{\{M\}_K\}_K$ are not considered cyclic. As we will see, the original result of Abadi and Rogaway does not apply to expressions with key cycles—a major weakness that we will correct in this work.

2.1.2 The AR Equivalence of Formal Expressions

A visible encryption term will appear 'opaque' to the adversary if and only if it is protected by at least one non-recoverable decryption key. Thus, we wish to say that two expressions are equivalent if they differ only in the contents of their 'opaque' encryption terms. To express this, Abadi and Rogaway define the *pattern* of an expression through which equivalence of expressions will be obtained:

Definition 2.4 (Pattern (Classical)). We define the *set of patterns*, **Pat**, by the grammar:

$$\textbf{Pat} ::= \textbf{Keys} \mid \textbf{Blocks} \mid (\textbf{Pat}, \textbf{Pat}) \mid \{\textbf{Pat}\}_{\textbf{Keys}} \mid \Box$$

The pattern of an expression M, denoted by *pattern*(M), is derived from M by replacing each encryption term $\{M'\}_K \in vis\,(M)$ (where $K \notin$ *R-Keys*(M)) by $\Box$

For two patterns P and Q, $P = Q$ is defined the following way:

- If $P \in$ **Blocks** $\cup$ **Keys**, then $P = Q$ iff P and Q are identical.
- If P is of the form $\Box$, then $P = Q$ iff Q is of the form $\Box$
- If P is of the form (P_1, P_2), then $P = Q$ iff Q is of the form (Q_1, Q_2) where $P_1 = Q_1$ and $P_2 = Q_2$.
- If P is of the form $\{P'\}_K$, then $P = Q$ iff Q is of the form $\{Q'\}_K$ where $P' = Q'$.

(Note that we call these 'classical' patterns. This is to distinguish them from the more complex patterns that we will consider later.)

One last complication remains before we can define formal equivalence. The first thing coming to mind is to say that two expressions are equivalent if their patterns are equal. However, consider two very simple formal expressions K_1 and K_2. Then these formal expressions would not be equivalent. On the other hand, these two expressions have the same meaning: a randomly drawn key. Despite being given different names, they both represent samples from the same distribution. It does not matter if we replace one of them with the other. More generally, we wish to formalise the notion of equivalence in such a way that renaming the keys yields in equivalent expression. Therefore, two formal expressions should be equivalent if their patterns differ only in the names of their keys.

Definition 2.5 (Key-Renaming Function). A bijection σ : **Keys** $\to$ **Keys** is called a *key-renaming function*. For any expression (or pattern) M, $M\sigma$ denotes the expression (or pattern) obtained from M by replacing all occurrences of keys K in M by $\sigma(K)$.

We are finally able to formalise the symbolic notion of equivalence:

Definition 2.6 (Equivalence of Expressions). We say that two expressions M and N are *equivalent*, denoted by $M \cong N$, if there exists a key-renaming function σ such that *pattern*$(M) =$ *pattern*$(N\sigma)$.

2.1.3 The Computational Model

The fundamental objects of the computational world are strings, $\mathbf{strings} = \{0,1\}^*$, and families of probability distributions over strings. These families are indexed by a *security parameter* $\eta \in \mathbf{parameters} = \mathbb{N}$ (which can be roughly understood as key-lengths). Two distribution families $\{D_\eta\}_{\eta\in\mathbb{N}}$ and $\{D'_\eta\}_{\eta\in\mathbb{N}}$ are *indistinguishable* if no efficient algorithm can determine from which distribution a value was sampled:

Definition 2.7 (Negligible Function). A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, written $f(n) \leq \operatorname{neg}(n)$, if for any $c > 0$ there is an $n_c \in \mathbb{N}$ such that $f(n) \leq n^{-c}$ whenever $n \geq n_c$.

Definition 2.8 (Indistinguishability). Two families $\{D_\eta\}_{\eta\in\mathbb{N}}$ and $\{D'_\eta\}_{\eta\in\mathbb{N}}$, are *indistinguishable*, written $D_\eta \approx D'_\eta$, if for all PPT adversaries A,

$$\left|\Pr\left[d \longleftarrow D_\eta; \mathsf{A}(1^\eta, d) = 1\right] - \Pr\left[d \longleftarrow D'_\eta; \mathsf{A}(1^\eta, d) = 1\right]\right| \leq \text{neg}\,(\eta)$$

In this model, pairing is an injective *pairing function* $[\cdot,\cdot] : \mathbf{strings} \times \mathbf{strings} \to \mathbf{strings}$ such that the length of the result only depends on the length of the paired strings. An encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ with key generation $\mathcal{K}$, encryption $\mathcal{E}$ and decryption $\mathcal{D}$. Let $\mathbf{plaintexts}$, $\mathbf{ciphertexts}$, and $\mathbf{keys}$ be nonempty subsets of $\mathbf{strings}$. The set $\mathbf{coins}$ is some probability field that stands for coin-tossing, *i.e.*, randomness.

Definition 2.9 (Symmetric Encryption Scheme). A *computational symmetric encryption scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K} : \mathbf{parameters} \times \mathbf{coins} \to \mathbf{keys}$ is a key-generation algorithm;
- $\mathcal{E} : \mathbf{keys} \times \mathbf{strings} \times \mathbf{coins} \to \mathbf{ciphertexts}$ is an encryption function;
- $\mathcal{D} : \mathbf{keys} \times \mathbf{strings} \to \mathbf{plaintexts}$ is such that for all $k \in \mathbf{keys}$ and $\omega \in \mathbf{coins}$,

$$\begin{aligned} \mathcal{D}(k, \mathcal{E}(k, m, \omega)) &= m \quad \text{for all } m \in \mathbf{plaintexts}, \\ \mathcal{D}(k, \mathcal{E}(k, m', \omega)) &= \perp \quad \text{for all } m' \notin \mathbf{plaintexts}. \end{aligned}$$

All of $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ are computable in polynomial-time in the length of the security parameter.

This definition, note, does not include any notion of security, and this must be defined separately. In fact, there are several different such definitions. Abadi and Rogaway, in their work, consider a spectrum of notions of their own devising, from 'type-0' to 'type-7.' Their main result uses the strongest of these notions, type-0:

Definition 2.10 (Type-0 Security). We say that a computational encryption scheme is type-0 secure if no probabilistic polynomial-time adversary A can distinguish the pair of oracles $(\mathcal{E}(k,\cdot), \mathcal{E}(k',\cdot))$ from the pair of oracles $(\mathcal{E}(k,0), \mathcal{E}(k,0))$ as k and k' are randomly generated. That is, for any probabilistic polynomial-time algorithm, A,

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot),\mathcal{E}(k',\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,0),\mathcal{E}(k,0)}(1^\eta) = 1\right] \leq \text{neg}\,(\eta)$$

Intuitively the above formula says the following: The adversary is given one of two pairs of oracles, either $(\mathcal{E}(k,\cdot), \mathcal{E}(k',\cdot))$ or $(\mathcal{E}(k,0), \mathcal{E}(k,0))$ (where the keys were randomly generated prior to handing the pair to the adversary), but it does not know which. Then, the adversary can perform any (probabilistic polynomial-time) computation, including several queries to the oracles. It can even query the oracles with messages that depend on previously given answers of the oracles. (The keys used by the oracles for encryption do not change while the adversary queries the oracles.) After this game, the adversary has to decide with which pair of oracles it was interacting. The adversary wins the game if he can decide for the correct one with a probability bigger than $\frac{1}{2}$, or (equivalently) if it can distinguish between the two. If this difference is negligible, as a function of η, we say the encryption scheme is type-0 secure.

As Abadi and Rogaway show, type-0 security is strong enough to provide *soundness* to the formal model. But to see this, we must first explain how the two models can be related.

2.1.4 The Interpretation Function, Soundness and Completeness

In order to prove any relationship between the formal and computational worlds, we need to define the *interpretation* of expressions and patterns. Once an encryption scheme is picked, we can define the interpretation function Φ, which assigns to each expression or pattern M a family of random variables $\{\Phi_\eta(M)\}_{\eta\in\mathbb{N}}$ such that each $\Phi_\eta(M)$ takes values in $\mathbf{strings}$. As in Abadi and Rogaway [AR02], this interpretation is defined in an algorithmic way. Intuitively,

- Blocks are interpreted as $\mathbf{strings}$,
- Each key is interpreted by running the key generation algorithm,
- Pairs are translated into computational pairs,
- Formal encryptions terms are interpreted by running the encryption algorithm on the interpretation of the plaintext and the interpretation of the key

For an expression M, we will denote by $[\![M]\!]_{\Phi_\eta}$ the distribution of $\Phi_\eta(M)$ and by $[\![M]\!]_\Phi$ the ensemble of $\{[\![M]\!]_{\Phi_\eta}\}_{\eta\in\mathbb{N}}$.

Then soundness and completeness are defined in the following way:

Definition 2.11 (Soundness (Classical)). We say that an interpretation is *sound in the classical sense*, or that an encryption scheme *provides classical soundness*, if the interpretation Φ (resulting from the encryption scheme) is such that for any given pairs of expressions M and N

$$M \cong N \Rightarrow [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

The primary result of Abadi and Rogaway given in [AR02] is that type-0 security provides classical soundness if the expressions M and N have no key-cycles.

Soundness has a counterpart, completeness. One can consider soundness to be the property that formal indistinguishability always becomes computational indistinguishability. One can think of completeness as the converse: computational indistinguishability is always the result of formal indistinguishability:

Definition 2.12 (Completeness (Classical)). We say that an interpretation is *complete* (in the classical sense), or that an encryption scheme *provides (classical) completeness*, if the interpretation Φ (resulting from the encryption scheme) is such that

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi \Rightarrow M \cong N$$

for any expressions M and N.

We remark that for the proofs of the soundness and completeness results, it was convenient for Abadi and Rogaway to introduce the interpretation of any pattern M (although this is not absolutely necessary). Therefore, boxes are interpreted as well, such that

- $\Box$ is interpreted by running the encryption algorithm on the fixed plaintext 0 and a randomly generated key.

The precise definition of $\Phi_\eta(M)$ for any pattern M is given by the algorithms in Figure 2.1. We note that these algorithms are fully defined for patterns, and because the grammar for patterns contains the grammar for expressions as a sub-grammar, they are fully defined for expressions as well.

```
algorithm INITIALIZE$(1^\eta, M)$
    for $K \in Keys(M)$ do $\tau(K) \longleftarrow \mathcal{K}(1^\eta)$
    let $k_0 \longleftarrow \mathcal{K}(1^\eta)$

algorithm CONVERT$(M)$
    if $M = K$ where $K \in$ Keys then
        return $\tau(K)$
    if $M = B$ where $B \in$ Blocks then
        return $B$
    if $M = (M_1, M_2)$ then
        $x \longleftarrow$ CONVERT$(M_1)$
        $y \longleftarrow$ CONVERT$(M_2)$
        return $[x, y]$
    if $M = \{M_1\}_K$ then
        $x \longleftarrow$ CONVERT$(M_1)$
        $y \longleftarrow \mathcal{E}(\tau(K), x)$
        return $y$
    if $M = \square$, then
        $y \longleftarrow \mathcal{E}(k_0, 0)$
        return $y$
```

Figure 2.1: Algorithmic components of the interpretation function

2.2 Soundness in the Presence of Key-Cycles

As we will see later, key-cycles do not cause a problem with completeness, however, as we discussed in the introduction, one of the weaknesses of the original Abadi-Rogaway's result is that it is not possible to prove soundness for expressions that included key-cycles. We will address this problem in this section starting by showing that, soundness in the presence of key-cycles is not possible to prove with the security notion adopted by Abadi and Rogaway. We suggest a new notion of security, KDM-security as a solution for the problem. In order to prove soundness, we will also need to extend our formal model, and after that we conclude this section showing that with this new definition of security it is possible to obtain soundness even in the presence of key-cycles.

2.2.1 Type-0 Security is Not Enough

In this section we show that type-0 security is not strong enough to ensure soundness in the case of key-cycles. That is, we demonstrate that it is possible to construct encryption schemes that are type-0, but fail to provide soundness in the presence of key-cycles.

Theorem 2.1. *Type-0 security does not imply soundness. That is, if there exists an encryption* *scheme that is type-0 secure, then there exists another encryption scheme which is also type-0 secure but does not provide soundness.*

Proof. This is shown via a simple counter-example. Assuming that there exists a type-0 secure encryption scheme, we will use it to construct another scheme which is also type-0 secure. However, we will show that this new scheme allows the adversary to distinguish one particular expression M from another particular expression N, even though $M \cong N$.

Let M be $\{K\}_K$ and let N be the expression $\{K_1\}_{K_2}$. Since these two expressions are equivalent, an encryption scheme that enforces soundness requires that the family of distributions:

$$\{k \longleftarrow \mathcal{K}(1^\eta); c \longleftarrow \mathcal{E}(k,k) : c\}_{\eta\in\mathbb{N}}$$

be indistinguishable from the family of distributions:

$$\{k_1 \longleftarrow \mathcal{K}(1^\eta); k_2 \longleftarrow \mathcal{K}(1^\eta); c \longleftarrow \mathcal{E}(k_1,k_2) : c\}_{\eta\in\mathbb{N}}.$$

However, this is not implied by Definition 2.10. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a type-0 secure encryption scheme. We assume that Π is such that keys and ciphertexts have different formats. Then, using Π, we construct a second type-0 secure encryption scheme $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ as follows:

- Let $\mathcal{K}' = \mathcal{K}$,
- Let $\mathcal{E}'$ be the following algorithm:

$$\mathcal{E}'(k,m) = \begin{cases} k & \text{if } m = k \\ \mathcal{E}(k,k) & \text{if } \mathcal{E}(k,m) = k \\ \mathcal{E}(k,m) & \text{otherwise} \end{cases}$$

- Let $\mathcal{D}'$ be the following algorithm:

$$\mathcal{D}'(k,c) = \begin{cases} k & \text{if } c = k \\ \mathcal{D}(k,k) & \text{if } c = \mathcal{E}(k,k) \\ \mathcal{D}(k,c) & \text{otherwise} \end{cases}$$

The scheme Π' acts exactly like Π unless the encryption algorithm $\mathcal{E}'$ is called on a pair (k,k). It is easy to see that this scheme is also type-0 secure.

To see this, suppose that Π' is not type-0 secure. That is, there exists some adversary A which can distinguish the pair of oracles $(\mathcal{E}'(k,\cdot), \mathcal{E}'(k',\cdot))$ from the pair $(\mathcal{E}'(k,0), \mathcal{E}'(k,0))$. There are two possibilities. Suppose that the adversary queried the oracle on k or k'. Then it would certainly be able to distinguish the oracle-pairs, but this also means that the adversary can produce the secret symmetric key to the scheme Π. Thus, the encryption scheme Π cannot be secure in any sense, much less type-0. Suppose, on the other hand, the adversary did not query the oracles on k or k' but managed to distinguish between the oracle pairs anyway. Then it was able to do so even though the encryption scheme Π' acted exactly like Π, and so Π cannot be type-0 secure.

Thus, the new scheme Π' must also be type-0 secure. However, it does not guarantee indistinguishability for the two distributions above. The first distribution will output always the encryption key while the second outputs a ciphertext, and these two distributions are easily distinguished by form alone. □

Remark 1. We note that in the proof, the expression M contains a key-cycle of length 1. What if all key-cycles are of length 2 or more? This question remains open. That is, there is no known type-0 secure encryption scheme which fails to provide soundness for key-cycles that are of length two or more.

Because type-0 encryption implies types 1 through 7, Theorem 2.1 implies that soundness with key-cycles cannot be provided by the security definitions devised by Abadi and Rogaway. In the next section, we show that this soundness property can, however, be met with *new* computational definitions.

2.2.2 KDM-Security

In the last section, we showed that the notions of security found in [AR00, AR02] are not strong enough to enforce soundness in the presence of key-cycles. However, *key-dependent message* (KDM) security, which was introduced by Black *et al.* [BRS02] (and in a weaker form by Camenisch and Lysyanskaya [CL01]), is strong enough to enforce soundness even in this case. (We note that Camenisch and Lysyanskaya also provided a natural application of KDM security, a credential system with interesting revocation properties, and so KDM security is of independent interest as well.)

KDM security both strengthens and weakens type-0 security. Recall that type-0 security allows the adversary to submit messages to an oracle which does one of two things:

- It could encrypt the message twice, under two different keys, or
- It could encrypt the bit 0 twice, under the same key.

An encryption scheme is type-0 secure if no adversary can tell which of these is being done. For KDM security, however, the game is slightly different. To over-simplify:

- The oracle in the KDM-security encrypts once, under only one key.
- Further, it encrypts either the message, or a *string* of 0's of equivalent length.
- However, it is willing to encrypt not just messages from the adversary, but also (more generally) *functions of the secret key*.

The first two of these differences make KDM security weaker than type-0 security. Specifically type-0 security conceals both the length of the plaintext and whether two ciphertext were created using the same encryption key or different ones. KDM security does not necessarily conceal either of these things. The last difference, however, is a significant strengthening. As its name suggests, KDM security remains strong even when the messages depend on the secret key—which, as Theorem 2.1 shows, is not necessarily true for type-0 security.

To provide the full picture, KDM security is defined in terms of *vectors* of keys and functions over these vectors. It is also defined in terms of oracles $\mathsf{Real}_{\bar{\mathbf{k}}}$ and $\mathsf{Fake}_{\bar{\mathbf{k}}}$, which work as follows:

- Suppose that for a fixed security parameter $\eta \in \mathbb{N}$, a vector of keys is given: $\bar{\mathbf{k}} = \{k_i \longleftarrow \mathcal{K}(1^\eta)\}_{i \in \mathbb{N}}$. (In each run of the key-generation algorithm independent coins are used.) The adversary can now query the oracles providing them with a pair (j, g), where $j \in \mathbb{N}$ and $g : \mathbf{keys}^\infty \rightarrow \{0,1\}^*$ is a constant length, deterministic function:
  - The oracle $\mathsf{Real}_{\bar{\mathbf{k}}}$ when receiving this input returns $c \longleftarrow \mathcal{E}(k_j, g(\bar{\mathbf{k}}))$;
  - The oracle $\mathsf{Fake}_{\bar{\mathbf{k}}}$ when receiving this same input returns $c \longleftarrow \mathcal{E}(k_j, 0^{|g(\bar{\mathbf{k}})|})$.

The challenge facing the adversary is to decide whether it has interacted with oracle $\mathsf{Real}_{\bar{\mathbf{k}}}$ or oracle $\mathsf{Fake}_{\bar{\mathbf{k}}}$. Formally:

Definition 2.13 (Symmetric-KDM Security). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Let the two oracles $\mathsf{Real}_{\bar{\mathbf{k}}}$ and $\mathsf{Fake}_{\bar{\mathbf{k}}}$ be as defined above. We say that the encryption scheme is *(symmetric) KDM-secure* if for all PPT adversaries A:

$$\Pr\left[\bar{\mathbf{k}} \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathsf{Real}_{\bar{\mathbf{k}}}}(1^\eta) = 1\right] - \Pr\left[\bar{\mathbf{k}} \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathsf{Fake}_{\bar{\mathbf{k}}}}(1^\eta) = 1\right] \leq \operatorname{neg}(\eta)$$

Remark 2. We note that although all known implementations of KDM-security are in the random-oracle model, this definition is well-founded even in the standard model. We also note that this definition is phrased in terms of indistinguishability. One could also imagine analogous definitions phrased in terms of non-malleability, but an exploration of those are beyond the scope of this dissertation.

We note that KDM-security implies type-3 security:

Definition 2.14 (Type-3 Security). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that the encryption-scheme is *type-3 secure* if no PPT adversary A can distinguish the oracles $\mathcal{E}(k, \cdot)$ and $\mathcal{E}(k, 0^{|\cdot|})$ as k is randomly generated, that is, for all PPT adversaries A:

$$\Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,0^{|\cdot|})}(1^\eta) = 1\right] \leq \operatorname{neg}(\eta)$$

In fact, the definition of type-3 encryption is exactly the same as that for KDM-security, except that the adversary must submit concrete messages to the encryption oracle instead of functions. But since the functions submitted in KDM security can be constant function that always produce a single output, the type-3 security ‘game’ is a special case of that for KDM security.

On the other hand, KDM security does not attempt to conceal the length of the plaintext (type-1 security) or that two ciphertexts were created with the same key (type-2 security). It will be impossible, therefore, for KDM security to provide soundness in the classical sense (Definition 2.11). Nonetheless, a weaker form of soundness can be achieved if the formal model is also weakened slightly.

2.2.3 A New Formal Model

In this section, we develop a weaker version of the formal model—one that allows formal encryption to leak partial information about the plaintext and key. One can think of this as a preview or special case of a later section, where we discuss such weakening in general (Section 2.5). In this section, however, we focus on the partial leakage allowed (in the computational model) by KDM security: the length of the plaintext, and whether two different ciphertexts were created using the same key.

To model the leakage of plaintext length, we first need to add the very concept of ‘length’ to the formal model:

Definition 2.15 (Formal Length). A formal length-function is a function symbol with fresh letter ℓ satisfying at least the following identities:

- For all blocks B_1 and B_2, $\ell(B_1) = \ell(B_2)$ iff $|B_1| = |B_2|$,
- For all expression M and key-renaming function σ, $\ell(M) = \ell(M\sigma)$,
- If $\ell(M_1) = \ell(N_1)$, $\ell(M_2) = \ell(N_2)$ then $\ell((M_1, M_2)) = \ell((N_1, N_2))$, and
- If $\ell(M) = \ell(N)$, then for all K_i, $\ell(\{M\}_{K_i}) = \ell(\{N\}_{K_i})$.

We would like to emphasise that these are the identities that a formal length function *minimally* has to satisfy. There may be more. In fact, if we only assume these properties, there is no hope to obtain completeness. We also remark, that it follows that for any key-renaming function σ, and expression M, $\ell(M) = \ell(M\sigma)$.

Given this, it is straightforward to add the required leakage to the formal model. If patterns represents those aspects of an expression that can be learned by the adversary, then patterns must now reveal the plaintext-length and key-names for undecryptable terms:

Definition 2.16 (Pattern (Type-3)). We define the *set of patterns*, **Pat**, by the grammar:

$$\textbf{Pat} ::= \textbf{Keys} \mid \textbf{Blocks} \mid (\textbf{Pat}, \textbf{Pat}) \mid \{\textbf{Pat}\}_{\textbf{Keys}} \mid \Box_{\textbf{Keys},\ell(\textbf{Exp})}$$

The type-3 pattern of an expression M, denoted by *pattern*$_3(M)$, is derived from M by replacing each encryption term $\{M'\}_K \in \mathit{vis}\,(M)$ (where $K \notin$ *R-Keys*(M)) by $\Box_{K,\ell(M')}$.

Note that the only difference between a type-3 pattern and a classical pattern is that an undecryptable term $\{M\}_K$ becomes $\Box_{K,\ell(M)}$ (*i.e.* labelled with the key and length) in type-3 patterns instead of merely $\Box$ in classical patterns.

Our notion of formal equality must be updated as well. For two patterns P and Q, $P \cong_3 Q$ is defined the following way:

Definition 2.17 (Formal Equivalence (Type-3)). We first introduce the relation $=_3$ between patterns:

- If $P \in \textbf{Blocks} \cup \textbf{Keys}$, then $P =_3 Q$ iff P and Q are identical.
- If P is of the form $\Box_{K,\ell(M')}$, then $P =_3 Q$ iff Q is of the form $\Box_{K,\ell(N')}$, and $\ell(M') = \ell(N')$ in the sense of Definition 2.15.

- If P is of the form (P_1, P_2), then $P =_3 Q$ iff Q is of the form (Q_1, Q_2) where $P_1 =_3 Q_1$ and $P_2 =_3 Q_2$.
- If P is of the form $\{P'\}_K$, then $P =_3 Q$ iff Q is of the form $\{Q'\}_K$ where $P' =_3 Q'$.

With this, we say that expressions M and N are *equivalent in the type-3 sense* (written $M \cong_3 N$) if there exists a key-renaming function σ such that *pattern*$_3(M) =_3$ *pattern*$_3(N\sigma)$. (Since a key-renaming function replaces all occurrences of K with $\sigma(K)$, we note that under σ, $\square_{K,\ell(M)}$ will become $\square_{\sigma(K),\ell(M\sigma)}$.)

Lastly, the above change to formal equivalence requires that the notions of soundness and completeness be similarly altered:

Definition 2.18 (Soundness (Type-3)). We say that an interpretation is *type-3 sound*, or that an encryption scheme *provides soundness in the type-3 sense*, if the interpretation Φ (resulting from the encryption scheme) is such that

$$M \cong_3 N \Rightarrow [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

for any pair of expressions M and N.

Definition 2.19 (Completeness (Type-3)). We say that an interpretation is *type-3 complete*, or that an encryption scheme *provides completeness in the type-3 sense*, if the interpretation Φ (resulting from the encryption scheme) is such that for any pair of expressions M and N,

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi \Rightarrow M \cong_3 N.$$

2.2.4 Soundness for Key-Cycles

Below, we present our two main soundness results: if an encryption scheme is KDM secure, it also provides type-3 soundness even in the presence of key-cycles.

Theorem 2.2 (Symmetric KDM Security Implies Soundness). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a computational symmetric encryption scheme such that $|\mathcal{E}(k, m, w)| = |\mathcal{E}(k, m, w')|$ for all $k \in \mathbf{keys}, m \in \mathbf{plaintexts}$ and $w, w' \in \mathbf{coins}$. Then, if the length-function ℓ satisfies only the equalities listed in Definition 2.15, and Π is KDM-secure, then Π provides type-3 soundness.*

Proof. We first redefine the interpretation of patterns. The only thing we have to change is the interpretation of a box. Now, the interpretation of a pattern $\square_{K,\ell(M)}$ for a given security parameter η is given by $\Phi_\eta(\{0^{|\Phi_\eta(M)|}\}_K)$. That is, the interpretation function used to encrypt a single 0 under a random key. Now, it encrypts a string of 0s of the same requisite length (length of $\Phi_\eta(M)$), and it encrypts them under the correct key $\tau(K)$.

The proof in this case is a somewhat reduced hybrid argument. In a standard hybrid argument, like the one Abadi and Rogaway used to prove their soundness result, several patterns are put between M and N; then, using security, it is proven that soundness holds between each two consecutive patterns, and therefore soundness holds for M and N. In our case, we first directly prove that $[\![M]\!]_\Phi$ is indistinguishable from $[\![$*pattern*$_3(M)]\!]_\Phi$. Then, since that holds for N too, and since *pattern*$_3(M)$ differs from *pattern*$_3(N)$ only in the name of keys, $[\![pattern_3(M)]\!]_\Phi$ is indistinguishable from $[\![pattern_3(N)]\!]_\Phi$, therefore the result follows. KDM security is used when we show that $[\![M]\!]_\Phi$ and $[\![pattern_3(M)]\!]_\Phi$ are indistinguishable.

For an arbitrary (formal) key K, let $\iota(K)$ denote the index of K. For an expression M, a set of formal (unrecoverable) keys S, and a function τ : **Keys** $\setminus S \to \mathbf{keys}$, we define a function $f_{M,S,\tau} : \mathbf{coins}^{e(M)} \times \mathbf{keys}^\infty \to \mathbf{strings}$ (where $e(M)$ is the number of encryptions in M) inductively in the following way:

- For $M = B \in$ **Blocks**, let $f_{B,S,\tau} : \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{B,S,\tau}(\bar{\mathbf{k}}) = B$;
- For $M = K \in$ **Keys** $\cap S$, let $f_{K,S,\tau} : \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{K,S,\tau}(\bar{\mathbf{k}}) = k_{\iota(K)}$;
- For $M = K \in$ **Keys** $\cap \overline{S}$, let $f_{K,S,\tau} : \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{K,S,\tau}(\bar{\mathbf{k}}) = \tau(K)$;
- For $M = (M_1, M_2)$, let $f_{(M_1,M_2),S,\tau} : \mathbf{coins}^{e(M_1)} \times \mathbf{coins}^{e(M_2)} \times \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{(M_1,M_2),S,\tau}(\omega_{M_1}, \omega_{M_2}, \bar{\mathbf{k}}) = [f_{M_1,S,\tau}(\omega_{M_1}, \bar{\mathbf{k}}), f_{M_2,S,\tau}(\omega_{M_2}, \bar{\mathbf{k}})]$;
- For $M = \{N\}_K$ and $K \in S$, let $f_{\{N\}_K,S,\tau} : \mathbf{coins} \times \mathbf{coins}^{e(N)} \times \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{\{N\}_K,S,\tau}(\omega, \omega_N, \bar{\mathbf{k}}) = \mathcal{E}(k_{\iota(K)}, f_{N,S,\tau}(\omega_N, \bar{\mathbf{k}}), \omega)$;
- For $M = \{N\}_K$ and $K \notin S$, let $f_{\{N\}_K,S,\tau} : \mathbf{coins} \times \mathbf{coins}^{e(N)} \times \mathbf{keys}^\infty \to \mathbf{strings}$ be defined as $f_{\{N\}_K,S,\tau}(\omega, \omega_N, \bar{\mathbf{k}}) = \mathcal{E}(\tau(K), f_{N,S,\tau}(\omega_N, \bar{\mathbf{k}}), \omega)$.

We note that this function is constant length because the keys are constant-length (for the same η) and the length of an encryption only depends on the length of the message and η.

We first prove that $[\![M]\!]_\Phi \approx [\![pattern_3(M)]\!]_\Phi$. Suppose that $[\![M]\!]_\Phi \not\approx [\![pattern_3(M)]\!]_\Phi$. This means that there is an adversary A that distinguishes the two distributions, that is

$$\Pr(x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1) - \Pr(x \longleftarrow [\![pattern_3(M)]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1)$$

is a non-negligible function of η. We will show that this contradicts the fact that the system is (symmetric) KDM-secure. To this end, we construct an adversary that can distinguish between the oracles $\mathsf{Real}_{\bar{\mathbf{k}}}$ and $\mathsf{Fake}_{\bar{\mathbf{k}}}$. From now on, let $S =$ **Keys** $\setminus$ *R-Keys*(M). Consider the following algorithm:

```
**algorithm** $\mathsf{B}^{\mathcal{F}}(1^\eta, M)$
    **for** $K \in$ *R-Keys*$(M)$ **do** $\tau(K) \longleftarrow \mathcal{K}(1^\eta)$
    $y \longleftarrow$ CONVERT2$(M, M)$
    $b \longleftarrow \mathsf{A}(1^\eta, y)$
    **return** $b$

**algorithm** CONVERT2$(M', M)$ with $M' \sqsubseteq M$
    **if** $M' = K$ where $K \in$ *R-Keys*$(M)$ **then**
        **return** $\tau(K)$
    **if** $M = B$ where $B \in$ **Blocks then**
        **return** $B$
```

```
**if** $M' = (M_1, M_2)$ **then**
    $x \longleftarrow \text{CONVERT2}(M_1, M)$
    $y \longleftarrow \text{CONVERT2}(M_2, M)$
    **return** $[x, y]$
**if** $M' = \{M_1\}_K$ with $K \in$ *R-Keys*$(M)$ **then**
    $x \longleftarrow \text{CONVERT2}(M_1, M)$
    $y \longleftarrow \mathcal{E}(\tau(K), x)$
    **return** $y$
**if** $M' = \{M_1\}_K$ with $K \notin$ *R-Keys*$(M)$ **then**
    $\omega \longleftarrow \mathbf{coins}^{e(M_1)}$
    $y \longleftarrow \mathcal{F}(\iota(K), f_{M_1,S,\tau}(\omega, .))$
    **return** $y$
```

This algorithm applies the distinguisher $\mathsf{A}(1^\eta, \cdot)$ on the distribution $[\![M]\!]_\Phi$ when $\mathcal{F}$ is $\mathsf{Real}_{\bar{\mathbf{k}}}$, and the distribution of $[\![\textit{pattern}_3(M)]\!]_\Phi$ when $\mathcal{F}$ is $\mathsf{Fake}_{\bar{\mathbf{k}}}$. So, if $\mathsf{A}(1^\eta, \cdot)$ can distinguish $[\![M]\!]_\Phi$ and $[\![\textit{pattern}_3(M)]\!]_\Phi$, then $\mathsf{B}^{\mathcal{F}}(1^\eta, \cdot)$ can distinguish $\mathsf{Real}_{\bar{\mathbf{k}}}$ and $\mathsf{Fake}_{\bar{\mathbf{k}}}$. But we assumed that $\mathsf{Real}_{\bar{\mathbf{k}}}$ and $\mathsf{Fake}_{\bar{\mathbf{k}}}$ cannot be distinguished, so $[\![M]\!]_\Phi \approx [\![\textit{pattern}_3(M)]\!]_\Phi$.

In a similar manner, we can show that $[\![N]\!]_\Phi \approx [\![\textit{pattern}_3(N)]\!]_\Phi$. Finally, it is easy to see that $[\![\textit{pattern}_3(M)]\!]_\Phi = [\![\textit{pattern}_3(N)]\!]_\Phi$, because the two patterns differ only by key-renaming. Hence $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. □

We conclude our consideration of KDM security by demonstrating what Black *et al.* claimed informally: the notion of KDM security is 'orthogonal' to the previous definitions of security. In particular, we claim that KDM security neither implies nor is implied by type-0 security. The former is proved directly, Theorem 2.4, while the latter is a corollary of previous theorems:

Corollary 2.3. *Type-0 security does not imply (symmetric) KDM-security. If there exists an encryption scheme that is type-0 secure, there exists an encryption scheme which is also type-0 secure but not KDM-secure.*

Proof. Suppose that there exists a type-0 secure encryption scheme. By Theorem 2.1 there is a type-0 secure scheme Π such that Π does not satisfy soundness. If all type-0 encryptions schemes are KDM-secure, then Π is as well. By Theorem 2.2, this means that Π satisfies soundness—a contradiction. □

Theorem 2.4. *KDM security does not imply type-0 security. That is, there is an encryption scheme that is KDM-secure, but not type-0 secure.*

Proof. This is easily seen by inspecting the KDM-secure encryption scheme given by Black *et al.* in the random oracle model [BRS02]. Let RO be the random oracle, which implements a random function from $\{0,1\}^*$ to $\{0,1\}^\infty$. Let $Pad \oplus M$ and $M \oplus Pad$ (where $M \in \{0,1\}^*$ and $Pad \in \{0,1\}^\infty$) be the bit-wise exclusive-or of M and the first $|M|$ bits of Pad. (Note that $|Pad \oplus M| = |M|$ exactly.) Let η be the security parameter. Then:

- $\mathcal{K}$ produces a random bit-string $K \longleftarrow \{0,1\}^\eta$.

- The encryption algorithm $\mathcal{E}$, on input (K, M), selects a random bit-string $r \longleftarrow \{0,1\}^\eta$ and returns the pair $(r, M \oplus RO(r||K))$.
- $\mathcal{D}$, on input $(K, C = (c_1, c_2))$, returns $c_2 \oplus RO\,(c_1||K)$.

This scheme is not type-0 secure because ciphertexts reveal the length of the plaintext. In particular, if c is a ciphertext for plaintext m, then $|c| = |m| + \eta$. Thus, one can easily distinguish between an oracle that encrypts the input message m and an oracle that always encrypts the 1-bit string 0. □

2.3 Partial Leakage of Information

In the previous section, we were forced by the definition of KDM security to consider encryption schemes that (possibly) revealed partial information about the plaintext (in particular its length) or the key (such as whether two ciphertexts were made using the same one). For the rest of this discussion, we leave behind the issue of key-cycles and concentrate our attention upon the issues of such partial leakage. In particular, we will eventually (Section 2.5) consider fully general notions of partial leakage. To motive these results, we first present soundness and completeness theorems for two specific examples. However, we do not prove them here, because they follow from the general treatment in Section 2.5, where we will return to these examples. In this section, we will separate the leakage of plaintext-length (type-1 encryption) from the leakage of key-sharing (type-2 encryption) and consider each separately. (We will also consider information-theoretic encryption schemes, but these we delay until Section 2.4.) In particular, we will show in this section that soundness can survive such leakage in the computational model if the formal model is appropriately weakened to match.

2.3.1 Soundness and Completeness for Type-1 Schemes

We start this discussion by considering the case of ‘type-1’ encryption schemes: encryption schemes which may reveal plaintext-length, but which conceals whether or not two ciphertexts were created using the same key. (In the terminology of Abadi and Rogaway, type-1 encryption is message-concealing and which-key concealing, but may be length-revealing.) An equivalent way to express this security definition is that no adversary should be able to tell whether two ciphertexts were created using the same key or different (independent) keys, even if the adversary is allowed to choose the plaintexts, so long as those plaintexts have the same length:

Definition 2.20 (Type-1 Security). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that the encryption-scheme is *type-1 secure* if no PPT adversary A can distinguish the pair of oracles $(\mathcal{E}(k,\cdot), \mathcal{E}(k',\cdot))$ and $(\mathcal{E}(k, 0^{|\cdot|}), \mathcal{E}(k, 0^{|\cdot|}))$ as k and k' are independently generated, that is, for all PPT adversaries A:

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot),\mathcal{E}(k',\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,0^{|\cdot|}),\mathcal{E}(k,0^{|\cdot|})}(1^\eta) = 1\right] \leq \operatorname{neg}(\eta)$$

Type-1 security does not provide soundness for the logic of Definition 2.1. For example, one can see immediately that $\{0\}_{K_1} \cong \{00\}_{K_1}$, but $[\![\{0\}_{K_1}]\!]_\Phi \not\approx [\![\{00\}_{K_1}]\!]_\Phi$ if the encryption scheme reveals the length of the plaintext.

To show soundness or completeness, patterns must reflect those aspects of an expression that an adversary can and cannot see. The idea is similar to the one in Definition 2.16, but now "boxes" are indexed with the only properties leaked by type-1 encryption: the formal length of the plaintext. (Note, however, that the notions of visible-subexpressions, recoverable keys and formal length remain unchanged.)

Definition 2.21 (Pattern (Type-1)). We define the *set of patterns*, **Pat**, by the grammar:

$$\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \square_{\ell(\mathbf{Exp})}$$

The type-1 pattern of an expression M, denoted by *pattern*$_1(M)$, is derived from M by replacing each term $\{M'\}_K \in vis(M)$ (where $K \notin$ *R-Keys*(M)) by $\square_{\ell(M')}$.

We say that two expressions M and N are ***type-1 equivalent***, and denote it by $M \cong_1 N$, if there exists a key-renaming function σ such that *pattern*$_1(M) =_1$ *pattern*$_1(N\sigma)$ where $=_1$ is defined in the following way:[1]

- If $P \in \mathbf{Blocks} \cup \mathbf{Keys}$, then $P =_1 Q$ iff P and Q are identical.
- If P is of the form $\square_{\ell(M')}$, then $P =_1 Q$ iff Q is of the form $\square_{\ell(N')}$, and $\ell(M') = \ell(N')$ in the sense of Definition 2.15.
- If P is of the form (P_1, P_2), then $P =_1 Q$ iff Q is of the form (Q_1, Q_2) where $P_1 =_1 Q_1$ and $P_2 =_1 Q_2$.
- If P is of the form $\{P'\}_K$, then $P =_1 Q$ iff Q is of the form $\{Q'\}_K$ where $P' =_1 Q'$.

Again, the symbol $\square_{\ell(M')}$ in a pattern reveals that some plaintext is encrypted and its length is $\ell(M')$.

Example 2.2. Let N be the expression

$$((\{0\}_{K_8}, \{100\}_{K_1}), ((K_7, \{(\{101\}_{K_9}, \{K_8\}_{K_5})\}_{K_5}), \{K_5\}_{K_7})).$$

We have that *R-Keys*$(N) = \{K_5, K_7, K_8\}$, and so, in this case, *pattern*$_1(N)$ is

$$((\{0\}_{K_8}, \square_{\ell(100)}), ((K_7, \{(\square_{\ell(101)}, \{K_8\}_{K_5})\}_{K_5}), \{K_5\}_{K_7})).$$

Defining M as in Example 2.1, *pattern*$_1(M)$ is

$$((\{0\}_{K_6}, \square_{\ell(\{K_7\}_{K_1})}), ((K_2, \{(\square_{\ell(001)}, \{K_6\}_{K_5})\}_{K_5}), \{K_5\}_{K_2})).$$

Now, if we replace $K_6 \to K_8$, $K_2 \to K_7$ and $K_5 \to K_5$ in M, we have that $M \cong_1 N$ iff $\ell(100) = \ell(\{K_7\}_{K_1})$.

[1]This notion is well-defined, since renaming a key inside an expression M does not affect $\ell(M)$, as we defined $\ell(K_i) = \ell(K_j)$ for all i, j.

With these definitions, the following soundness and completeness theorems can be proved. Since these theorems are simply special cases of the general soundness and completeness theorems in Section 2.5, the proofs are deferred until later (Examples 2.12 and 2.22).

Theorem 2.5 (Type-1 Soundness). *Let Π be a type-1 secure encryption scheme such that for all $k \in \mathbf{keys}, m \in \mathbf{plaintexts}$ and $w, w' \in \mathbf{coins}$ we have $|\mathcal{E}(k,m,w)| = |\mathcal{E}(k,m,w')|$. Then, if the length-function satisfies only the equalities defined in Definition 2.15, then for any M and N expressions such that B-Keys(M) and B-Keys(N) are not cyclic in M and N respectively,*

$$M \cong_1 N \textit{ implies } [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

Otherwise, for arbitrary length-function ℓ (that is, one satisfying possible more equations), if for all pairs of expressions M and N, $\ell(M) = \ell(N)$ implies that the binary length of $[\![M]\!]_{\Phi_\eta}$ is the same as the binary length of $[\![N]\!]_{\Phi_\eta}$ for each security parameter η, then for any M and N expressions,

$$M \cong_1 N \textit{ implies } [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

In addition to soundness, we also demonstrate completeness. If soundness shows that formal indistinguishability implies computational indistinguishability, completeness shows the converse. Rephrased, completeness implies that formal *distinguishability* (as opposed to *in*distinguishability) implies computational distinguishability. For this to be true, the interpretation function must enforce a handful of 'atomic' distinguishability properties:

Theorem 2.6 (Type-1 Completeness). *Let Π be a type-1 secure encryption scheme such that $|\mathcal{E}(k,m,w)| = |\mathcal{E}(k,m,w')|$ for all $k \in \mathbf{keys}, m \in \mathbf{plaintexts}$ and $w, w' \in \mathbf{coins}$. We have that,*

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi \textit{ implies } M \cong_1 N$$

*for all M and N pairs of expressions if and only if the following conditions hold: for any $K, K', K'' \in$ **Keys**, $B \in$ **Blocks**, $M, M', N \in$ **Exp**,*

(i) *no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$,*

(ii) *if $[\![(K,\{M\}_K)]\!]_\Phi \approx [\![(K'',\{M'\}_{K'})]\!]_\Phi$, then $K' = K''$, and*

(iii) *if $[\![\{M\}_K]\!]_\Phi \approx [\![\{M'\}_{K'}]\!]_\Phi$ then $\ell(M) = \ell(M')$.*

Some aspects of this theorem merit further discussion. First, note that the theorem does not mention key-cycles. Secondly, note that Condition (i) requires that different types of objects, blocks, keys, pairs and encryption terms should be distinguishable to achieve completeness; this can be ensured by tagging each object with its type, as suggested in [AR02]. Thirdly, Condition (ii) (which we call *weak confusion-freeness*) is equivalent to the property of weak key-authenticity introduced by Horvitz and Gligor [HG03] in the case of type-0 schemes. This property essentially means that decrypting with the wrong key should be detectable in a probabilistic sense. Finally, condition (iii) requires that encryption of messages with different length should be detectable. Definition 2.20 *allows* that encryptions of messages of different length may be detected but does not enforce it. That suffices for soundness, but completeness requires that it should be detectable when ciphertexts contain messages of different lengths. A purely computational condition that implies condition (iii) is the notion of *strictly length revealing*:

Definition 2.22 (Strictly Length Revealing Scheme). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that the encryption-scheme is *strictly length revealing* if it is type-1 secure but there exists a PPT adversary A such that the following function is a non-negligible function of η:

$$\Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,0^{\neq|\cdot|})}(1^\eta) = 1\right]$$

We use $0^{\neq|\cdot|}$ to denote 0^n, where $n \neq |\cdot|$.

2.3.2 Soundness and Completeness for Type-2 Schemes

Having considered the leakage of plaintext-length in the previous section, we turn to the other to the kinds of leakage seen in KDM-security: whether or not two ciphertext share a key. However, we now assume that the plaintext conceals the plaintext-length. ('Type-2' in the terminology of Abadi and Rogaway, as well as message-concealing, length-concealing, and which-key revealing.) For this type of encryption, no adversary should be able to tell whether a ciphertext contains a (possibly long) plaintext or the single-bit plaintext 0:

Definition 2.23 (Type-2 Security). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that the encryption-scheme is *type-2 secure* if no PPT adversary A can distinguish the oracles $\mathcal{E}(k,\cdot)$ and $\mathcal{E}(k,0)$ as k is randomly generated, that is, for all PPT adversaries A:

$$\Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,0)}(1^\eta) = 1\right] \leq \mathrm{neg}\,(\eta)$$

Again, patterns must be re-defined to reflect all the information about an expression which may be available to the adversary, but only that information:

Definition 2.24 (Pattern (Type-2)). We define the *set of patterns*, **Pat**, by the grammar:

$$\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \Box_{\mathbf{Keys}}$$

The type-2 pattern of an expression M, denoted by *pattern*$_2(M)$, is derived from M by replacing each term $\{M'\}_K \in \mathit{vis}\,(M)$ (where $K \notin$ *R-Keys*(M)) by $\Box_K$.

We say that two expressions M and N are *type-2 equivalent*, and denote it by $M \cong_2 N$, if there exists a key-renaming function σ such that *pattern*$_2(M) =_2$ *pattern*$_2(N\sigma)$ where $=_2$ is defined in the following way:[2]

- If $P \in \mathbf{Blocks} \cup \mathbf{Keys}$, then $P =_2 Q$ iff P and Q are identical.
- If P is of the form $\Box_K$, then $P =_2 Q$ iff Q is also of the form $\Box_K$.

[2]The key-renaming function σ affects all the occurrences of a key K, including those occurrences as indexes of $\Box$.

- If P is of the form (P_1, P_2), then $P =_2 Q$ iff Q is of the form (Q_1, Q_2) where $P_1 =_2 Q_1$ and $P_2 =_2 Q_2$.
- If P is of the form $\{P'\}_K$, then $P =_2 Q$ iff Q is of the form $\{Q'\}_K$ where $P' =_2 Q'$.

Example 2.3. Let N be the same expression as in Example 2.2,

$$((\{0\}_{K_8}, \{100\}_{K_1}), ((K_7, \{(\{101\}_{K_9}, \{K_8\}_{K_5})\}_{K_5}), \{K_5\}_{K_7})).$$

We have that *R-Keys*$(N) = \{K_5, K_7, K_8\}$, and so, in this case, *pattern*$_2(N)$ is

$$((\{0\}_{K_8}, \Box_{K_1}), ((K_7, \{(\Box_{K_9}, \{K_8\}_{K_5})\}_{K_5}), \{K_5\}_{K_7})).$$

Defining M as in Example 2.1, *pattern*$_2(M)$ is

$$((\{0\}_{K_6}, \Box_{K_4}), ((K_2, \{(\Box_{K_3}, \{K_6\}_{K_5})\}_{K_5}), \{K_5\}_{K_2})).$$

Now, if we replace $K_6 \to K_8$, $K_4 \to K_1$, $K_2 \to K_7$, $K_3 \to K_9$ and $K_5 \to K_5$ in M, we have that $M \cong_2 N$.

With these definitions, the following soundness and completeness theorems can be proved. Again, since these theorems are special cases of our general soundness and completeness theorems in Section 2.5, the proofs are deferred until later (Examples 2.13 and 2.22).

Theorem 2.7 (Type-2 Soundness). *Let M and N be expressions such that B-Keys(M) and B-Keys(N) are not cyclic in M and N respectively, and Π a type-2 secure encryption scheme. Then,*

$$M \cong_2 N \textit{ implies } [\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

Theorem 2.8 (Type-2 Completeness). *Let Π be a type-2 secure encryption scheme. We have that,*

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi \textit{ implies } M \cong_2 N$$

*for any pairs of expressions M and N if and only if the following conditions hold: for any $K, K', K'' \in$ **Keys**, $B \in$ **Blocks**, $M, M', N, N' \in$ **Exp**,*

(i) *no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$,*

(ii) *if $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$,*

(iii) *if $[\![(\{M\}_K, \{M'\}_K)]\!]_\Phi \approx [\![(\{N\}_{K'}, \{N'\}_{K''})]\!]_\Phi$ then $K' = K''$.*

The conditions of the completeness theorem are similar to the ones for the type-1 case except for condition (iii). This condition requires that encryption with different keys should be detectable. Definition 2.23 *allows* that encrypting with different keys may be detectable, but it does not *require* it. That suffices for soundness, but such detection *is* required for completeness. It is easily shown that condition (iii) is implied by the purely computational definition of a *strictly key revealing* encryption scheme:

Definition 2.25 (Strictly Key Revealing Scheme). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that the encryption-scheme is *strictly key revealing* if it is type-2 secure but there exists a PPT adversary A such that the following function is a non-negligible function of η:

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot),\mathcal{E}(k',\cdot)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot),\mathcal{E}(k,\cdot)}(1^\eta) = 1\right]$$

2.3.3 Soundness and Completeness for Type-3 Schemes

Type-3 encryption schemes (Definition 2.14, also called message-concealing, which-key revealing and length-revealing in the terminology of Abadi and Rogaway) can be thought of as leaking the information leaked by both type-1 and type-2 schemes. Both soundness and completeness results follow using the notion of patterns from Definition 2.16. As with type-1 and type-2 encryption, completeness requires that it is possible to distinguish ciphertexts that were encrypted with different keys, and to distinguish ciphertexts for which the plaintexts have different lengths. That is, the encryption scheme must be both strictly key revealing *and* strictly length revealing (Definitions 2.25 and 2.22 respectively).

2.4 Information-Theoretic Interpretations: Soundness and Completeness for One-Time Pad

Besides the computational definition, there are other possible important notions of 'indistinguishability.' For example, we could say that two distributions are 'indistinguishable' if and only if they are *identical*. Such a notion would lead to new (but analogous) notions of soundness and completeness, and we can explore these new notions using (as a specific encryption scheme) the One-Time Pad (OTP).

Let $\mathbf{strings} := \{0,1\}^*$ with the following pairing function: For any two strings $x, y \in \mathbf{strings}$ we can define the pairing of x and y as $[x, y] := \langle x, y, 0, 1_{|y|}\rangle$ where $\langle\ ,\ ,\ \dots\ ,\ \rangle$ denotes the concatenation of the strings separated by the commas, 1_m stands for m many 1's, and for any $x \in \{0,1\}^*$, $|x|$ denotes the length of the string. The number of 1's at the end indicate how long the second string is in the pair, and the 0 separates the strings from the 1's. Let $\mathbf{blocks}$ be those strings that end with 001. The ending is just a tag, it shows that the meaning of the string is a block.

Key-Generation. In case of the OTP, the length of the encrypting key must match the length of the plaintext. Thus, we need a separate key-generation for each length. That is, for each $n > 3$, $\mathcal{K}_n$ is a random variable over some discrete probability field $(\Omega_{\mathcal{K},n}, \Pr_{\mathcal{K},n})$ such that its values are equally distributed over $\mathbf{keys}_n := \{k \mid k \in \mathbf{strings},\ |k| = n,\ k \text{ ends with } 010\}$. Let $\mathbf{keys} := \bigcup_4^\infty \mathbf{keys}_n$. For $k \in \mathbf{keys}$, let $\mathrm{core}(k)$ denote the string that we get from k by cutting the tag 010.

Encryption. Let the domain of the encryption function, $\mathrm{Dom}_{\mathcal{E}}$, be those elements $(k, x) \in \mathbf{keys} \times \mathbf{strings}$, for which $|k| = |x| + 3$, and let $\mathcal{E}(k, x) := \langle \mathrm{core}(k) \oplus x, 110\rangle$. The tag 110 informs us that the string is a ciphertext. Notice that this encryption is not probabilistic, hence $\mathcal{E}(k, x)$ is not a random variable. Notice also, that the tag of the plaintext is not dropped, that part is also encrypted.

Decryption. The decryption function $\mathcal{D}(k, x)$ is defined whenever $|k| = |x|$, and, naturally the value of $\mathcal{D}(k, x)$ is the first $|k| - 3$ bits of $k \oplus x$.

Indistinguishability. As we mentioned, let us now call two distributions indistinguishable, if they are identical, and denote this relation by $=_d$.

As in the case of type-3 encryption, lengths of the messages are revealed. Therefore, we must again define the *length* of an expression.

Definition 2.26. We assume that some length function l : **Keys** $\rightarrow \{4, 5, ...\}$ is given on the keys symbols. The length of a block is defined as $l(B) := |B| + 3$. We added 3 to match the length of the tag. We define the length function on any expression in **Exp** by induction:

- $l((M, N)) := l(M) + 2l(N) + 1$,
- $l(\{M\}_K) := l(M) + 3$, if $l(M) = l(K) - 3$, and
- $l(\{M\}_K) := 0$, if $l(M) \neq l(K) - 3$.

The *valid expressions* are defined as those expressions in which the length of the encrypted subexpressions match the length of the encrypting key, and, in which no key is used twice to encrypt. (This latter condition is necessary to prevent leaking information because of the properties of the OTP.)

Definition 2.27. We define the *valid expressions for OTP* as $\mathbf{Exp}_{\mathrm{OTP}} = \{M \in \mathbf{Exp} \mid M' \sqsubseteq M$ implies $l(M') > 0$, and each key encrypts at most once in $M\}$.

The interpretation function for the OTP is defined similarly to the other cases, with some minor changes regarding the tagging of the messages. Also, there is no security parameter in this encryption scheme, so the interpretation outputs a single random variable for each formal expression (rather than a family of such variables). We present here the full algorithm:

```
algorithm INTERPRETATION$_{\mathrm{OTP}}(M)$
    for $K \in Keys(M)$ do $\tau(K) \longleftarrow \mathcal{K}_{l(K)}$
    $y \longleftarrow$ CONVERT$_{\mathrm{OTP}}(M)$
    return $y$

algorithm CONVERT$_{\mathrm{OTP}}(N)$
    if $N = K$ where $K \in$ Keys then
        return $\tau(K)$
    if $N = B$ where $B \in$ Blocks then
        return $\langle B, 100\rangle$
    if $N = (N_1, N_2)$ then
        return $[\mathrm{CONVERT}_{\mathrm{OTP}}(N_1), \mathrm{CONVERT}_{\mathrm{OTP}}(N_2)]$
    if $N = \{N_1\}_K$ then
        return $\langle \mathcal{E}(\tau(K), \mathrm{CONVERT}_{\mathrm{OTP}}(N_1)), 110\rangle$
```

As in the previous cases, we must again find a suitable equivalence relation for formal expressions. One possibility is to index the boxes again with the encrypting keys. Another possibility is to label the boxes with the length as well, but in the OTP scheme, the key reveals the length of the ciphertext. Therefore, we can use the first, that is a simpler possibility. Thus OTP-patterns are defined as follows:

Definition 2.28 (Pattern (OTP)). We define the *set of patterns,* **Pat**, by the grammar:

$$\textbf{Pat} ::= \textbf{Keys} \mid \textbf{Blocks} \mid (\textbf{Pat}, \textbf{Pat}) \mid \{\textbf{Pat}\}_{\textbf{Keys}} \mid \Box_{\textbf{Keys}}$$

The OTP pattern of a valid expression M, denoted by $\textit{pattern}_{\text{OTP}}(M)$, is derived from M by replacing each term $\{M'\}_K \in \textit{vis}\,(M)$ (where $K \notin \textit{R-Keys}(M)$) by $\Box_K$.

We say that two expressions M and N are *OTP equivalent*, and denote it by $M \cong_{\text{OTP}} N$, if there exists a length-preserving key-renaming function σ such that $\textit{pattern}_{\text{OTP}}(M) =_2 \textit{pattern}_{\text{OTP}}(N\sigma)$ with $=_2$ as in Definition 2.24

Then, then following soundness and completeness theorems can be proved. Again, these theorems are special cases of the general soundness and completeness theorems in Section 2.5. Thus, the proofs will be deferred (Examples 2.14 and 2.23, respectively).

Theorem 2.9 (OTP Soundness). *Let M and N be two valid expressions in **Exp**$_{\text{OTP}}$ such that B-Keys(M) and B-Keys(N) are not cyclic in M and N respectively. Then,*

$M \cong_{OTP} N$ implies that $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$ are the same probability distributions.

Theorem 2.10 (OTP Completeness). *Let M and N be two valid expressions in **Exp**$_{\text{OTP}}$. Then*

if $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$ have the same probability distributions, we have that $M \cong_{OTP} N$.

Note that the completeness theorem for OTP does not contain any side conditions like those of Theorems 2.6 and 2.8. This is because here, what would have been condition (i) from Theorem 2.6 is immediate due to the tagging. The natural condition (ii) also follows from the tagging since decrypting with the wrong key will result in a meaningless text. Lastly, the natural Condition (iii) is meaningless in this case since we just encrypt at most once with each key.

2.5 A General Treatment for Symmetric Encryption

In this section, we provide a general treatment of soundness and completeness for the Abadi-Rogaway type logics of formal encryptions. The following contain the cases discussed in the previous two sections as special cases. In Subsection 2.5.1 we present a general probabilistic framework for symmetric encryptions, which includes both the computational and the information-theoretic encryption schemes. Then, in Subsection 2.5.2, we show a general way to handle partial leaking of encryption in the formal view. This will be done essentially via an equivalence relation on the set of encryption terms, which is meant to express which encryption terms are indistinguishable for an adversary. In that section, we also introduce the important notion that we call *properness* of this equivalence relation. This is essential, because this is exactly the property that will make an Abadi-Rogaway type hybrid argument go through. Finally, in the remaining subsections of this section, we present the interpretation, the general soundness and completeness results as well as discussions of how the theorems for the type-1, type-2 and OTP cases that we presented before follow from the general theorems.

2.5.1 A General Treatment for Symmetric Encryptions

We provide a general probabilistic framework for symmetric encryption, which contains both the computational and the information-theoretic description as special cases. Keys, plaintexts and ciphertexts are elements of some discrete set $\overline{\mathbf{strings}}$. This is $(\{0,1\}^*)^\infty$ in the case of a computational treatment, and it is $\{0,1\}^*$ for the information-theoretic description. The elements of $(\{0,1\}^*)^\infty$ are sequences in $\{0,1\}^*$, corresponding to a parameterisation by the security parameter.

A fixed subset, $\overline{\mathbf{plaintext}} \subseteq \overline{\mathbf{strings}}$ represents the messages that are allowed to be encrypted. Another subset, $\overline{\mathbf{keys}} \subseteq \overline{\mathbf{strings}}$ is the possible set encrypting keys that corresponds to the range of the key generation algorithm $\mathcal{K}$. In order to be able to build up longer messages from shorter ones, we assume that an injective *pairing function* is given: $[\,.\,,\,.\,] : \overline{\mathbf{strings}} \times \overline{\mathbf{strings}} \to \overline{\mathbf{strings}}$. The range of the pairing function will be called $\overline{\mathbf{pairs}}$: $\overline{\mathbf{pairs}} := \mathrm{Ran}_{[\,.\,,\,.\,]}$. A symmetric encryption scheme has the following constituents:

Key-generation. Key-generation is represented by a random variable $\mathcal{K} : \Omega_{\mathcal{K}} \to \overline{\mathbf{keys}}$, over a discrete probability field $(\Omega_{\mathcal{K}}, \Pr_{\mathcal{K}})$. In a given scheme, more than one key-generation is allowed.

Encryption. For a given $k \in \overline{\mathbf{keys}}$, and a given $x \in \overline{\mathbf{plaintext}}$, $\mathcal{E}(k,x)$ is a random variable over some discrete probability field $(\Omega_{\mathcal{E}}, \Pr_{\mathcal{E}})$. The values of this random variable are in $\overline{\mathbf{strings}}$ and are denoted by $\mathcal{E}(k,x)(\omega)$, whenever $\omega \in \Omega_{\mathcal{E}}$.

Decryption. An encryption must be decryptable, so we assume that for each $k \in \overline{\mathbf{keys}}$, a function $\mathcal{D} : (k,x) \mapsto \mathcal{D}(k,x)$ is given satisfying $\mathcal{D}_k\big(\mathcal{E}(k,x)(\omega)\big) = x$ for all $\omega \in \Omega_{\mathcal{E}}$ and $x \in \overline{\mathbf{plaintext}}$.

The notion of *indistinguishability* is important both in case of computational and information-theoretic treatments of cryptography. It expresses when there is only very small probability to tell two probability distributions apart.

Indistinguishability. We assume that an equivalence relation called *indistinguishability* is defined on distributions over $\overline{\mathbf{strings}}$. We will denote this relation by $\approx$. We will also say that two random variables taking values in $\overline{\mathbf{strings}}$ are equivalent (indistinguishable) if (and only if) their distributions are equivalent; we will use $\approx$ for denoting this equivalence between random variables as well. For $\approx$, we require the followings:

(i) Random variables with the same distribution are indistinguishable;

(ii) Constant random variables are indistinguishable if and only if the constants are the same;

(iii) For random variables $F : \Omega_F \to \overline{\mathbf{strings}}$ and $G : \Omega_G \to \overline{\mathbf{strings}}$, if $F \approx G$, the following must hold: If π^i denotes the projection onto one of the components of $\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}$, then $\pi^i \circ [\cdot,\cdot]^{-1} \circ F \approx \pi^i \circ [\cdot,\cdot]^{-1} \circ G$ for $i = 1, 2$;

(iv) If $F' : \Omega_F \to \overline{\mathbf{strings}}$, $G' : \Omega_G \to \overline{\mathbf{strings}}$ are also indistinguishable random variables such that F and F' are independent and G and G' are also independent, then $\omega_F \mapsto [F(\omega_F), F'(\omega_F)]$ and $\omega_G \mapsto [G(\omega_G), G'(\omega_G)]$ are indistinguishable random variables; moreover, if $\alpha, \beta : \overline{\mathbf{strings}} \to \overline{\mathbf{strings}}$ are functions that preserve $\approx$ (*i.e.* $\alpha \circ F \approx \alpha \circ G$

and $\beta \circ F \approx \beta \circ G$ whenever $F \approx G$), then $\omega_F \mapsto [(\alpha \circ F)(\omega_F), (\beta \circ F)(\omega_F)]$ and $\omega_G \mapsto [(\alpha \circ G)(\omega_G), (\beta \circ G)(\omega_G)]$ are indistinguishable random variables if $F \approx G$.

Indistinguishability needs to satisfy some further properties under encryption and decryption that we will specify under the definition of encryption schemes below.

Example 2.4. The simplest example for indistinguishability is that it holds between two random variables if and only if their distributions are identical.

Example 2.5. The standard notion of computational indistinguishability in [Yao82] is also a special case of the general definition. In this case $\overline{\mathbf{strings}} = (\{0,1\}^*)^\infty = \mathbf{strings}^\infty$. Random variables of computational interest have the form $F : \Omega_F \to \mathbf{strings}^\infty$ and have independent components; *i.e.*, for $\eta \in \mathbb{N}$ security parameter, denoting the η'th component of F by $F_\eta : \Omega_F \to \mathbf{strings}$, it is required that F_η and $F_{\eta'}$ are independent random variables for $\eta \neq \eta'$. Indistinguishability then is phrased with the ensemble of probability distributions of the components of the random variables.

Definition 2.29. An *encryption scheme* is a quadruple $\Pi = (\{\mathcal{K}_i\}_{i \in I}, \mathcal{E}, \mathcal{D}, \approx)$ where $\{\mathcal{K}_i\}_{i \in I}$ is a set of key-generations for some index set I, $\mathcal{E}$ is an encryption, $\mathcal{D}$ decrypts ciphertexts encrypted by $\mathcal{E}$, and $\approx$ is the indistinguishability defined above. We require that for any $i \in I$, the probability distribution of $\mathcal{K}_i$ be distinguishable from any constant in $\overline{\mathbf{strings}}$, the distributions of $\mathcal{K}_i$ and of $\mathcal{K}_j$ be distinguishable whenever $i \neq j$, and also that the distribution of (k, k') be distinguishable from the distribution of (k, k) if k and k' are independently generated: $k \longleftarrow \mathcal{K}_i$, $k' \longleftarrow \mathcal{K}_j$ for any $i, j \in I$. The indistinguishability relation $\approx$, besides satisfying the properties stated before, needs to be such that if F and G are random variables taking values in $\overline{\mathbf{strings}}$, and $\mathcal{K}_i$ is a key-generation such that the distribution of $[\mathcal{K}_i, F]$ is indistinguishable from the distribution of $[\mathcal{K}_i, G]$, then:

(i) $(\omega_\mathcal{E}, \omega_{\mathcal{K},i}, \omega) \mapsto \mathcal{E}\big(\mathcal{K}_i(\omega_{\mathcal{K},i}), F(\omega)\big)(\omega_\mathcal{E})$ and $(\omega_\mathcal{E}, \omega_{\mathcal{K},i}, \omega) \mapsto \mathcal{E}\big(\mathcal{K}_i(\omega_{\mathcal{K},i}), G(\omega)\big)(\omega_\mathcal{E})$ are indistinguishable random variables;

(ii) $(\omega_{\mathcal{K},i}, \omega) \mapsto \mathcal{D}\big(\mathcal{K}_i(\omega_{\mathcal{K},i}), F(\omega)\big)$ and $(\omega_{\mathcal{K},i}, \omega) \mapsto \mathcal{D}\big(\mathcal{K}_i(\omega_{\mathcal{K},i}), G(\omega)\big)$ are also indistinguishable random variables.

Here the probability over $\Omega_{\mathcal{K}_i} \times \Omega_F$ is the joint probability of $\mathcal{K}_i$ and F, which are here not necessarily independent. Similarly for G.

2.5.2 Equivalence of Expressions

In their treatment, Abadi and Rogaway defined equivalence of expressions via replacing encryption terms encrypted with non-recoverable keys in an expression by a box; two expressions then were declared equivalent if once these encryption terms were replaced, the obtained *patterns* looked the same up to *key-renaming*. This method implicitly assumes, that an adversary cannot distinguish any undecryptable terms. However, if we want to allow leakage of partial information, we need to modify the notion of equivalence.

Before introducing our notion of equivalence of expressions, we postulate an equivalence notion $\equiv_{\mathbf{K}}$ on the set of keys, and another equivalence, $\equiv_{\mathbf{C}}$ on the set of *valid* encryption terms. The word *valid*, defined precisely below, is meant for those encryption terms (and expressions) that "make sense". Then, the equivalence on the set of valid expressions will be defined with the help of $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$.

The reason for postulating equivalence on the set of keys is that we want to allow many key-generation processes in the probabilistic setting. We therefore have to be able to distinguish formal keys that were generated by different key-generation processes. Therefore, we assume that an equivalence relation $\equiv_{\mathbf{K}}$ is given on the set of keys such that each equivalence class contains infinitely many keys. Let $\mathcal{Q}_{\mathbf{Keys}} := \mathbf{Keys}\big/\equiv_{\mathbf{K}}$.

Definition 2.30 (Key-Renaming Function). A bijection $\sigma : \mathbf{Keys} \rightarrow \mathbf{Keys}$ is called *key-renaming function*, if $\sigma(K) \equiv_{\mathbf{K}} K$ for all $K \in \mathbf{Keys}$. For any expression M, $M\sigma$ denotes the expression obtained from M by replacing all occurrences of keys K in M by $\sigma(K)$.

Definition 2.31. We define the *support* of a key-renaming function σ, and denote it by *supp*(σ), as the subset of **Keys** such that $\sigma(K) \neq K$.

We say that two key-renaming functions σ and τ are *compatible* if for all keys $K \in$ *supp*$(\sigma) \cap$ *supp*(τ) we have that $\sigma(K) = \tau(K)$.

The set **Exp** is often too big to suit our purposes. For example, sometimes we require that certain messages can be encrypted with certain keys only. We therefore define the set of valid expressions:

Definition 2.32. A set of *valid expressions* is a subset $\mathbf{Exp}_{\mathcal{V}}$ of **Exp** such that:

(i) all keys and all blocks are contained in $\mathbf{Exp}_{\mathcal{V}}$;

(ii) if $M \in \mathbf{Exp}_{\mathcal{V}}$ then *sub*$(M) \subset \mathbf{Exp}_{\mathcal{V}}$ and any number of pairs of elements in *sub*(M) are also in $\mathbf{Exp}_{\mathcal{V}}$;

(iii) for any key-renaming function σ, $M \in \mathbf{Exp}_{\mathcal{V}}$ iff $M\sigma \in \mathbf{Exp}_{\mathcal{V}}$.

Given a set of valid expressions, the set of *valid encryption terms* is $\mathbf{Enc}_{\mathcal{V}} := \mathbf{Enc} \cap \mathbf{Exp}_{\mathcal{V}}$.

Equivalence of valid expressions is meant to incorporate the notion of security into the model: we want two expressions to be equivalent when they look the same to an adversary. If we think that the encryption is so secure that no partial information is revealed, then all undecryptable terms should look the same to an adversary. If partial information, say repetition of the encrypting key, or length is revealed, then we have to adjust the notion of equivalence accordingly. We do this by introducing an equivalence relation on the set of valid encryption terms in order to capture which ciphertexts an adversary can and cannot distinguish; in other words, what partial information (length, key, etc...) can an adversary retrieve from the ciphertext.

Hence, we will assume that there is an equivalence relation, $\equiv_{\mathbf{C}}$ given on the set of valid encryption terms, with the property that for any $M, N \in \mathbf{Enc}_{\mathcal{V}}$ and σ key-renaming function, $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$. Let $\mathcal{Q}_{\mathbf{Enc}} := \mathbf{Enc}_{\mathcal{V}}\big/\equiv_{\mathbf{C}}$.

Since we required that $M \equiv_{\mathbf{C}} N \in \mathbf{Enc}_{\mathcal{V}}$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$ whenever σ is a key-renaming function, σ induces a renaming on $\mathcal{Q}_{\mathbf{Enc}}$, which we also denote by σ.

Example 2.6 (Length-Revealing). In Section 2.3.1 two encryption terms were considered to be indistinguishable for an adversary if and only if they had the same length. In this case, we define $\equiv_{\mathbf{C}}$ so that it equates encryption terms with the same length, and hence an element of $\mathcal{Q}_{\mathbf{Enc}}$ will contain all encryption terms that have a specific length.

Example 2.7 (Which-Key Revealing). In Section 2.3.2 we considered the situation when an adversary can recognise that two encryption terms were encrypted with different keys. For this case, we will need to define $\equiv_{\mathbf{C}}$ so that two encryption terms are equivalent if and only if they are encrypted with the same key.

Definition 2.33 (Formal Logic of Symmetric Encryption). A formal logic for symmetric encryption is a triple $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ where $\mathbf{Exp}_{\mathcal{V}}$ is a set of valid expressions, $\equiv_{\mathbf{K}}$ is an equivalence relation on **Keys**, and $\equiv_{\mathbf{C}}$ is an equivalence relation on $\mathbf{Enc}_{\mathcal{V}}$; we require the elements of $\mathcal{Q}_{\mathbf{Keys}}$ to be infinite sets, and that for any σ key renaming function relative to $\mathcal{Q}_{\mathbf{Keys}}$,

(i) if $M \in \mathbf{Exp}$, then $M \in \mathbf{Exp}_{\mathcal{V}}$ if and only if $M\sigma \in \mathbf{Exp}_{\mathcal{V}}$;

(ii) if $M, N \in \mathbf{Enc}_{\mathcal{V}}$, then $M \equiv_{\mathbf{C}} N$ if and only if $M\sigma \equiv_{\mathbf{C}} N\sigma$;

(iii) replacing an encryption term within a valid expression with another equivalent valid encryption term results in a valid expression.

To define the equivalence of expressions, we first assign to each valid expression an element in the set of *patterns*, **Pat**, defined the following way:

Definition 2.34 (Pattern). We define the set of *patterns*, **Pat**, by the grammar:

$$\mathbf{Pat} ::= \mathbf{Keys} \mid \mathbf{Blocks} \mid (\mathbf{Pat}, \mathbf{Pat}) \mid \{\mathbf{Pat}\}_{\mathbf{Keys}} \mid \Box_{\mathcal{Q}_{\mathbf{Enc}}}$$

The pattern of a valid expression M, denoted by *pattern*(M), is obtained from M by replacing each undecryptable term $\{M'\}_K \sqsubseteq M$ ($K \notin$ *R-Keys*(M)) by $\Box_{\mu(\{M'\}_K)}$, where $\mu(\{M'\}_K) \in \mathcal{Q}_{\mathbf{Enc}}$ denotes the equivalence class containing $\{M'\}_K$.

Definition 2.35 (Equivalence of Expressions). We say that two valid expressions M and N are *equivalent*, and denote it by $M \cong N$, if there exists a key-renaming function σ such that *pattern*$(M) =$ *pattern*$(N\sigma)$, where for any pattern Q, $Q\sigma$ denotes the pattern obtained by renaming all the keys and the box-indexes (which are equivalence classes in $\mathcal{Q}_{\mathbf{Enc}}$) in Q with σ.

Example 2.8. In the case when the elements of $\mathcal{Q}_{\mathbf{Enc}}$ contain encryption terms encrypted with the same key, Example 2.7, there is a one-to-one correspondence between $\mathcal{Q}_{\mathbf{Enc}}$ and **Keys**, and therefore we can index the boxes with keys instead of the elements in $\mathcal{Q}_{\mathbf{Enc}}$: $\Box_K$, $K \in \mathbf{Keys}$. Then if N is the same expression as in Example 2.3, the pattern according to the above definition is the same as we had in that example. In that example M and N are equivalent according to the definition of equivalence above.

Proper Equivalence of Ciphers

In order to make the soundness and completeness proofs work, we need to have some restrictions on $\equiv_{\mathbf{C}}$; without any restrictions, the proofs will never work. The condition that we found the most natural for our purposes is what we call *proper equivalence,* defined below. This condition will make soundness work. For completeness, besides proper equivalence, we need to assume something for the relationship of $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$. We call our assumption *independence,* and it is defined in Definition 2.37. Let us start by defining the set μ_{key}, for each $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, as

$$\mu_{\text{key}} := \{K \in \textbf{Keys} \mid \text{there is a valid expression } M \text{ such that } \{M\}_K \in \mu\}.$$

Definition 2.36 (Proper Equivalence of Ciphers). We say that an equivalence relation $\equiv_{\mathbf{C}}$ on $\mathbf{Enc}_{\mathcal{V}}$ is *proper*, if for any finite set of keys S, if $\mu \in \mathcal{Q}_{\mathbf{Enc}}$ contains an element of the form $\{N\}_K$ with $K \notin S$, we have that:

1. if $|\mu_{\text{key}}|$ is finite then μ also contains an element C such that *Keys*$(C) \cap S = \emptyset$, and $K \not\sqsubseteq C$;
2. if $|\mu_{\text{key}}| = \infty$ then μ also contains an element C such that *Keys*$(C) \cap (S \cup \{K\}) = \emptyset$.

In other words, if μ contains an element encrypted with a key K not in S, then μ has a representative in which no key of S appears, and in which K may only appear as an encrypting key, but not as a subexpression, or in the case of a class with infinitely many encrypting keys there is an element in which no keys from $S \cup \{K\}$ appear. In fact, we show in Proposition 2.11 that the cardinality of the set μ_{key} is equal to either 1 or ∞.

Example 2.9. If $\equiv_{\mathbf{C}}$ denotes the equivalence of Example 2.7 (*i.e.* two ciphers are equivalent iff they have the same encrypting key, hence $|\mu_{\text{key}}| = 1$), then it is clearly proper, since if $\{M\}_K \in \mu$, and $K \notin S$, then $C = \{K'\}_K$ works for any $K' \notin S$; there is such a K', since we assumed that there are infinitely many keys. $C = \{B\}_K$ ($B \in$ **Blocks**) is also a good choice since **Blocks** is not empty.

Example 2.10. If $\equiv_{\mathbf{C}}$ denotes the equivalence of Example 2.6, then it is clearly proper ($|\mu_{\text{key}}| = \infty$). If $\{M\}_K \in \mu$, $K \notin S$, then $C = \{M'\}_{K'}$ is a good choice where C is constructed by assigning to each key in $\{M\}_K$, a new key K'' not in $S \cup \{K\}$. We can do this since we assumed that there are infinitely many keys. Then, since key-renaming does not change the length, $\ell(M) = \ell(M')$, and μ contains all encryption terms of the same length, $C \in \mu$ and properness follows.

The following propositions will be useful for proving our general soundness and completeness results.

Proposition 2.11. *Let $\Delta = (\boldsymbol{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. Then, the equivalence relation $\equiv_{\mathbf{C}}$ is such that for any equivalence class $\mu \in \mathcal{Q}_{\boldsymbol{Enc}}$, μ_{key} has either one, or infinitely many elements.*

Proof. Let $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, and assume that there are more than one encrypting key in μ_{key} (but $|\mu_{\text{key}}|$ finite), that is, there are two different keys K and K_1 such that $\{M\}_K, \{M_1\}_{K_1} \in \mu$ for some valid expressions M and M_1. Since $\equiv_{\mathbf{C}}$ is proper and $\{M_1\}_{K_1} \in \mu$, if we consider $S = \{K\}$ ($K_1 \neq K$ thus $K_1 \notin S$) then μ has an element of the form $\{M'\}_{K'}$ in which no key of S appears and in which K_1 may only appear as an encrypting key, but not as a subexpression. In particular we have that

$$K \notin \mathit{Keys}(M') \text{ and } K \neq K'. \tag{2.1}$$

Since we assumed that each equivalence class in $\mathcal{Q}_{\mathbf{Keys}}$ contains infinitely many elements (recall Definition 2.33), there is a key $L \neq K$ such that $L \equiv_{\mathbf{K}} K$, and

$$L \notin \mathit{Keys}(\{M\}_K) \cup \mathit{Keys}(\{M'\}_{K'}). \tag{2.2}$$

Then, defining σ to do nothing else but to switch the keys L and K, we have using (2.2) that

$$\{M\}_K\sigma = \{M\sigma\}_L$$

and (by (2.1) and (2.2))

$$\{M'\}_{K'}\sigma = \{M'\}_{K'}.$$

But, since $\{M\}_K \equiv_{\mathbf{C}} \{M'\}_{K'}$, we have (by definition of formal logic) that

$$\{M\}_K\sigma \equiv_{\mathbf{C}} \{M'\}_{K'}\sigma$$

that is

$$\{M\sigma\}_L \equiv_{\mathbf{C}} \{M'\}_{K'}.$$

Since $\{M'\}_{K'} \in \mu$, it must hold that $\{M\sigma\}_L \in \mu$. Therefore, there are infinitely many encrypting keys in μ since there are infinitely many choices for L. □

Proposition 2.12. *Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. If σ is a key-renaming function (relative to $\equiv_{\mathbf{K}}$), then for any $\mu \in \mathcal{Q}_{\mathbf{Enc}}$, $|\mu_{\text{key}}| = |\sigma(\mu)_{\text{key}}|$.*

Proof. If $|\mu_{\text{key}}| = \infty$, then $|\sigma(\mu)_{\text{key}}| = \infty$, since for any $\{M\}_K \in \mu$, $\{M\}_K\sigma = \{M\sigma\}_{\sigma(K)} \in \sigma(\mu)$. Since σ is a bijection, and since any μ contains either only one or infinitely many elements, the claim follows. □

The meaning of the next proposition is that if $\equiv_{\mathbf{C}}$ is proper, then given a set of valid ciphers $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ such that none of the encrypting keys are in S, and if $\mu_1, ...\mu_l$ are all the equivalence classes of the elements in $\mathfrak{C}$, then it is possible to choose a representative of each of μ_j, denoted by C_{μ_j}, such that no key of S occurs in any of C_{μ_j}, none of the L_i's occur as a subexpression in any C_{μ_j}, and no key occurs in two of C_{μ_j} unless the corresponding two equivalence classes both have only the same, single encrypting key.

Proposition 2.13. *Let $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be such that $\equiv_{\mathbf{C}}$ is proper. Let $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ be a set of valid encryption terms, and S a finite set of keys with $L_i \notin S$ ($i \in \{1, ..., n\}$). Let $\mu(\mathfrak{C})$ denote the set of all equivalence-classes with respect to $\equiv_{\mathbf{C}}$ of all elements in $\mathfrak{C}$. Then, for each $\nu \in \mu(\mathfrak{C})$, there is an element $C_\nu \in \nu$ such that:*

(i) *Keys*$(C_\nu) \cap S = \emptyset$

(ii) $L_i \not\sqsubseteq C_\nu$ *for all* $i \in \{1, ..., n\}$

(iii) *if* $\nu \neq \nu'$, $|\nu_{\text{key}}| \neq \infty$ *and* $|\nu'_{\text{key}}| \neq \infty$, *then Keys*$(C_\nu) \cap$ *Keys*$(C_{\nu'}) \neq \emptyset$ *if and only if* $\nu_{\text{key}} = \nu'_{\text{key}} = \{K\}$ *for some key* K*, and in this case*

1. *Keys*$(C_\nu) \cap$ *Keys*$(C_{\nu'}) = \{K\}$,
2. C_ν *and* $C_{\nu'}$ *are both of the form* $\{\cdot\}_K$ *with the same* K*, and*
3. $K \not\sqsubseteq C_\nu$, $K \not\sqsubseteq C_{\nu'}$.

(iv) *if* $\nu \neq \nu'$ *and either* $|\nu_{\text{key}}| = \infty$ *or* $|\nu'_{\text{key}}| = \infty$, *then Keys*$(C_\nu) \cap$ *Keys*$(C_{\nu'}) = \emptyset$.

Proof. Observe, that if μ_i denotes the equivalence class of $\{N_i\}_{L_i}$ in $\mathcal{Q}_{\textbf{Enc}}$, then $\nu \in \mu(\mathfrak{C})$ if and only if $\nu = \mu_i$ for some $i \in \{1, ...n\}$. Proof goes by induction.

The statement is clearly true if $n = 1$, since $\equiv_{\mathbf{C}}$ is proper.

Suppose now that the result is true for $n-1$. Let $\{N_1\}_{L_1}$, $\{N_2\}_{L_2}$,..., $\{N_n\}_{L_n}$ be valid expressions, and let S be a set of keys such that $L_i \notin S$. Without loss of generality, we can assume, that the numbering is such that there is an l, $1 \leq l \leq n$, such that $|(\mu_i)_{\text{key}}| = 1$ if $i \leq l$ and $|(\mu_i)_{\text{key}}| = \infty$ if $i > l$.

Case 1: Let us first assume that $l = n$, i.e., $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$, and that there is an $m \in \{1, ..., n-1\}$ such that $L_n = L_m$. Since the statement is assumed to be true for $n-1$, we have that for the family of encryption terms $\mathfrak{C}' = \{\{N_i\}_{L_i}\}_{i=1}^{n-1}$ and the set S we can choose C_{μ_i} for all $i \leq n-1$ such that conditions (i′), (ii′), (iii′) and (iv′) hold for these , that is,

(i′) *Keys*$(C_{\mu_i}) \cap S = \emptyset$ for all $1 \leq i \leq n-1$,

(ii′) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n-1$, and

(iii′) if $\mu_i \neq \mu_j$, $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) \neq \emptyset$ if and only if $(\mu_i)_{\text{key}} = (\mu_j)_{\text{key}} = \{K\}$ for some key K, and in that case

1. *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) = \{K\}$,
2. C_{μ_i} and C_{μ_j} are both of the form $\{\cdot\}_K$ with the same K, and
3. $K \not\sqsubseteq C_{\mu_i}$, $K \not\sqsubseteq C_{\mu_j}$.

(iv′) if $\mu_i \neq \mu_j$ and either $|(\mu_i)_{\text{key}}| = \infty$ or $|(\mu_j)_{\text{key}}| = \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) = \emptyset$.

We can immediately discard (iv′) since we suppose that $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$. Suppose now that $\mu_n = \mu_i$ for some $i \leq n-1$, then there is nothing to prove, $C_{\mu_n} = C_{\mu_i}$ has already been chosen and so (i), (ii) and (iii) are obviously satisfied by IH.

If there is no such i, then consider

$$S_{n-1} := \left(\left(\bigcup_{i=1}^{n-1} \textit{Keys}(C_{\mu_i}) \cup \{L_i\} \right) \setminus \{L_n\} \right) \cup S.$$

Since $\equiv_{\mathbf{C}}$ is proper (using S_{n-1} and $\{N_n\}_{L_n} \in \mu_n$), there is a $C \in \mu_n$ such that *Keys*$(C) \cap S_{n-1} = \emptyset$ and $L_n \not\sqsubseteq C$. Let us define $C_{\mu_n} = C$. Then:

(i) *Keys*$(C) \cap S = \emptyset$ follows from the fact that *Keys*$(C) \cap S_{n-1} = \emptyset$ and $S \subseteq S_{n-1}$;

(ii) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n$ since:

  (a) $L_i \not\sqsubseteq C_{\mu_j}$, for all $1 \leq i, j \leq n-1$ by (ii′),

  (b) $L_n \not\sqsubseteq C_{\mu_j}$, $1 \leq j \leq n-1$ because we assumed that $L_n = L_m$ and $L_m \not\sqsubseteq C_{\mu_j}$ by (ii′),

  (c) $L_i \not\sqsubseteq C$, for all $L_i \neq L_m$ such that $1 \leq i \leq n-1$ (remember that $L_n = L_m$) since $L_i \in S_{n-1}$ and *Keys*$(C) \cap S_{n-1} = \emptyset$, and

  (d) $L_n \not\sqsubseteq C$ by the way that C was chosen (hence $L_m \not\sqsubseteq C$).

(iii) (a) for all $1 \leq i, j \leq n-1$ it is true by (iii′);

  (b) Suppose now that $\mu_n \neq \mu_k$ and *Keys*$(C) \cap$ *Keys*$(C_{\mu_k}) \neq \emptyset$ for some $1 \leq k \leq n-1$. If we combine these with the fact that $Keys(C) \cap S_{n-1} = \emptyset$, we need to have that

$$\textit{Keys}(C) \cap \textit{Keys}(C_{\mu_k}) = \{L_n\}.$$

  It is now easy to see from the equation above that C and C_{μ_k} are both of the form $\{\cdot\}_{L_n}$. For that notice that by (ii.d) just proved above, $L_n \not\sqsubseteq C$ and by (ii.a) $L_n \not\sqsubseteq C_{\mu_k}$. The only thing left to show is that $(\mu_n)_{\text{key}} = (\mu_k)_{\text{key}} = \{L_n\}$. This comes straightforward from the fact that C and C_{μ_k} are both of the form $\{\cdot\}_{L_n}$ and from the fact that $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$. Combining these we have

$$(\mu_n)_{\text{key}} = (\mu_k)_{\text{key}} = \{L_n\}.$$

  The converse is very simple. Suppose that $(\mu_n)_{\text{key}} = (\mu_k)_{\text{key}} = \{L_n\}$. Since $C \in \mu_n$ and $C_{\mu_k} \in \mu_k$ we have that both are of the form $\{\cdot\}_{L_n}$ and thus *Keys*$(C) \cap$ *Keys*$(C_{\mu_k}) \neq \emptyset$. The rest follows as above.

(iv) Verified since by hypothesis we suppose that $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$.

Case 2: Suppose now that $l = n$, but there is no $m \in \{1, ..., n-1\}$ such that $L_n = L_m$. Since the result is true for $n-1$, we have that for the family of encryption terms $\mathfrak{C}' = \{\{N_i\}_{L_i}\}_{i=1}^{n-1}$ and the set $S' = S \cup \{L_n\}$ (note that $L_i \notin S'$ for all $i \leq n-1$) we can choose C_{μ_i} for all $i \leq n-1$ such that conditions (i′), (ii′), (iii′) and (iv′) hold for these , that is,

(i′) *Keys*$(C_{\mu_i}) \cap (S \cup \{L_n\}) = \emptyset$ for all $1 \leq i \leq n-1$,

(ii′) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n-1$, and

(iii′) if $\mu_i \neq \mu_j$, $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) \neq \emptyset$ if and only if $(\mu_i)_{\text{key}} = (\mu_j)_{\text{key}} = \{K\}$ for some key K;

(iv′) if $\mu_i \neq \mu_j$ and either $|(\mu_i)_{\text{key}}| = \infty$ or $|(\mu_j)_{\text{key}}| = \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) = \emptyset$.

Again, if $\mu_n = \mu_i$ for some $i < n$, then there is nothing to prove, let $C_{\mu_n} = C_{\mu_i}$ and note that (i) and (iii) are obviously satisfied, and (ii) ($L_n \not\sqsubseteq C_{\mu_j}$, for all $1 \leq j \leq n$, and $L_i \not\sqsubseteq C_{\mu_n}$ for all $1 \leq i \leq n-1$) follows from (i′) and (ii′) respectively. Again (iv) is also true since we suppose that $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$.

If there is no such i, then consider

$$S_{n-1} = \left(\bigcup_{i=1}^{n-1} \textit{Keys}(C_{\mu_i}) \cup \{L_i\} \right) \cup S.$$

By properness (using S_{n-1} and $\{N_n\}_{L_n} \in \mu_n$), and since $L_n \notin S_{n-1}$ (by (i′), assumption $L_n \neq L_i$ for all $i < n$, and by hypothesis of the proposition $L_n \notin S$), there is a $C \in \mu_n$ such that *Keys*$(C) \cap S_{n-1} = \emptyset$, and $L_n \not\sqsubseteq C$. Let us define $C_{\mu_n} = C$. Then:

(i) follows from (i′) and from the fact that *Keys*$(C) \cap S_{n-1} = \emptyset$;

(ii) is true, since:

(a) $L_i \not\sqsubseteq C_{\mu_j}$, for all $1 \leq i, j \leq n-1$ by (ii′),

(b) $L_n \not\sqsubseteq C_{\mu_j}$, for all $1 \leq j \leq n-1$ by (i′),

(c) $L_i \not\sqsubseteq C$ for $1 \leq i \leq n-1$ because by properness *Keys*$(C) \cap S_{n-1} = \emptyset$, and

(d) $L_n \not\sqsubseteq C$ because of properness.

(iii) follows, because

(a) for all $1 \leq i, j \leq n-1$ it is true by (iii′), and

(b) for the other case it holds since by properness *Keys*$(C) \cap S_{n-1} = \emptyset$ and thus *Keys*$(C) \cap$ *Keys*$(C_{\mu_i}) = \emptyset$ for all $1 \leq i \leq n-1$.

(iv) Verified since by hypothesis we suppose that $|(\mu_i)_{\text{key}}| = 1$ for all $1 \leq i \leq n$.

Case 3: Suppose now that $l < n$, but there is $m \in \{1, ..., n-1\}$ such that $L_n = L_m$. Since the result is assumed to be true for $n-1$, we have that for the family of encryption terms $\mathfrak{C}' = \{\{N_i\}_{L_i}\}_{i=1}^{n-1}$ and the set S we can choose C_{μ_i} for all $i \leq n-1$ such that conditions (i′), (ii′), (iii′) and (iv′) hold for these , that is,

(i′) *Keys*$(C_{\mu_i}) \cap S = \emptyset$ for all $1 \leq i \leq n-1$,

(ii′) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n-1$, and

(iii′) if $\mu_i \neq \mu_j$, $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) \neq \emptyset$ if and only if $(\mu_i)_{\text{key}} = (\mu_j)_{\text{key}} = \{K\}$ for some key K;

(iv′) if $\mu_i \neq \mu_j$ and either $|(\mu_i)_{\text{key}}| = \infty$ or $|(\mu_j)_{\text{key}}| = \infty$, then *Keys*$(C_{\mu_i}) \cap$ *Keys*$(C_{\mu_j}) = \emptyset$.

Again, suppose now that $\mu_n = \mu_i$ for some $i \leq n-1$, then there is nothing to prove, $C_{\mu_n} = C_{\mu_i}$ has already been chosen and so (i), (ii), (iii) and (iv) are obviously satisfied by IH.

If there is no such i, then consider

$$S_{n-1} := \left(\left(\bigcup_{i=1}^{n-1} \mathit{Keys}(C_{\mu_i}) \cup \{L_i\} \right) \setminus \{L_n\} \right) \cup S.$$

Since $\equiv_{\mathbf{C}}$ is proper (using S_{n-1} and $\{N_n\}_{L_n} \in \mu_n$, $|(\mu_n)_{\text{key}}| = \infty$), there is a $C \in \mu_n$ such that $\mathit{Keys}(C) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$. Then:

(i) $\mathit{Keys}(C) \cap S = \emptyset$ follows from the fact that $\mathit{Keys}(C) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$ and $S \subseteq (S_{n-1} \cup \{L_n\})$;

(ii) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n$ since:

   (a) $L_i \not\sqsubseteq C_{\mu_j}$, for all $1 \leq i, j \leq n-1$ by (ii′),

   (b) $L_n \not\sqsubseteq C_{\mu_j}$, $1 \leq j \leq n-1$ because we assumed that $L_n = L_m$ and $L_m \not\sqsubseteq C_{\mu_j}$ by (ii′),

   (c) $L_i \not\sqsubseteq C$, for all $L_i \neq L_m$ such that $1 \leq i \leq n-1$ (remember that $L_n = L_m$) since $L_i \in S_{n-1}$ and $\mathit{Keys}(C) \cap S_{n-1} = \emptyset$, and

   (d) $L_n \not\sqsubseteq C$ because $\mathit{Keys}(C) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$ (hence $L_m \not\sqsubseteq C$).

(iii) note that if $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$ then $1 \leq i, j \leq l < n$ and thus by HI (iii) holds.

(iv) (a) for the case $l \leq j \leq n-1$ and $1 \leq i \leq n-1$, $\mathit{Keys}(C_{\mu_j}) \cap \mathit{Keys}(C_{\mu_i}) = \emptyset$ holds by IH;

   (b) it is only left to show that for all $1 \leq i \leq n-1$, $\mathit{Keys}(C_{\mu_n}) \cap \mathit{Keys}(C_{\mu_i}) = \emptyset$. This is true because by definition $\mathit{Keys}(C_{\mu_n}) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$ and $\mathit{Keys}(C_{\mu_i}) \subseteq (S_{n-1} \cup \{L_n\})$.

Case 4: The proof of the remaining case, $l < n$, i.e., $|(\mu_i)_{\text{key}}| = \infty$ for $l < i \leq n$, and there is no $m \in \{1, ..., n-1\}$ such that $L_n = L_m$ is a combination of the proofs of Case 2 and Case 3. Since the result is true for $n-1$, we have that for the family of encryption terms $\mathfrak{C}' = \{\{N_i\}_{L_i}\}_{i=1}^{n-1}$ and the set $S' = S \cup \{L_n\}$ (note that $L_i \notin S'$ for all $i \leq n-1$) we can choose C_{μ_i} for all $i \leq n-1$ such that conditions (i′), (ii′), (iii′) and (iv′) hold for these , that is,

(i′) $\mathit{Keys}(C_{\mu_i}) \cap (S \cup \{L_n\}) = \emptyset$ for all $1 \leq i \leq n-1$,

(ii′) $L_i \not\sqsubseteq C_{\mu_j}$ for all $1 \leq i, j \leq n-1$, and

(iii′) if $\mu_i \neq \mu_j$, $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$, then $\mathit{Keys}(C_{\mu_i}) \cap \mathit{Keys}(C_{\mu_j}) \neq \emptyset$ if and only if $(\mu_i)_{\text{key}} = (\mu_j)_{\text{key}} = \{K\}$ for some key K;

(iv′) if $\mu_i \neq \mu_j$ and either $|(\mu_i)_{\text{key}}| = \infty$ or $|(\mu_j)_{\text{key}}| = \infty$, then $\mathit{Keys}(C_{\mu_i}) \cap \mathit{Keys}(C_{\mu_j}) = \emptyset$.

Again, if $\mu_n = \mu_i$ for some $i < n$, then there is nothing to prove, let $C_{\mu_n} = C_{\mu_i}$ and note that (i), (iii) and (iv) are obviously satisfied, and (ii) ($L_n \not\sqsubseteq C_{\mu_j}$, for all $1 \leq j \leq n$, and $L_i \not\sqsubseteq C_{\mu_n}$ for all $1 \leq i \leq n-1$) follows from (i′) and (ii′) respectively.

If there is no such i, then consider

$$S_{n-1} = \left(\bigcup_{i=1}^{n-1} \textit{Keys}(C_{\mu_i}) \cup \{L_i\} \right) \cup S.$$

By properness (using S_{n-1} and $\{N_n\}_{L_n} \in \mu_n$, $|(\mu_n)_{\text{key}}| = \infty$), and since $L_n \notin S_{n-1}$ (by (i′), assumption $L_n \neq L_i$ for all $i < n$, and by hypothesis of the proposition $L_n \notin S$), there is a $C \in \mu_n$ such that $\textit{Keys}(C) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$. Let us define $C_{\mu_n} = C$. Then:

(i) follows from (i′) and from the fact that $\textit{Keys}(C) \cap S_{n-1} = \emptyset$;

(ii) is true, since:

   (a) $L_i \not\sqsubseteq C_{\mu_j}$, for all $1 \leq i, j \leq n-1$ by (ii′),
   (b) $L_n \not\sqsubseteq C_{\mu_j}$, for all $1 \leq j \leq n-1$ by (i′),
   (c) $L_i \not\sqsubseteq C$ for $1 \leq i \leq n-1$ because by properness $\textit{Keys}(C) \cap S_{n-1} = \emptyset$, and
   (d) $L_n \not\sqsubseteq C$ because by definition of C, $\textit{Keys}(C) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$.

(iii) note that if $|(\mu_i)_{\text{key}}| \neq \infty$ and $|(\mu_j)_{\text{key}}| \neq \infty$ then $1 \leq i, j \leq l < n$ and thus by HI (iii) holds.

(iv) (a) for the case $l \leq j \leq n-1$ and $1 \leq i \leq n-1$, $\textit{Keys}(C_{\mu_j}) \cap \textit{Keys}(C_{\mu_i}) = \emptyset$ holds by IH;
   (b) it is only left to show that for all $1 \leq i \leq n-1$, $\textit{Keys}(C_{\mu_n}) \cap \textit{Keys}(C_{\mu_i}) = \emptyset$. This is true because by definition $\textit{Keys}(C_{\mu_n}) \cap (S_{n-1} \cup \{L_n\}) = \emptyset$ and $\textit{Keys}(C_{\mu_i}) \subseteq S_{n-1}$.

□

Given sets $\mathfrak{C}$ and S as in the conditions of the proposition, let $\mathfrak{R}(\mathfrak{C}, S)$ denote the nonempty set

$$\mathfrak{R}(\mathfrak{C}, S) := \left\{ \{C_\nu\}_{\nu \in \mu(\mathfrak{C})} \;\middle|\; \begin{array}{l} C_\nu \in \nu, \text{ and } \{C_\nu\}_{\nu\in\mathfrak{C}} \text{ and } S \text{ satisfy conditions} \\ \text{(i), (ii), (iii), and (iv) of Proposition 2.13} \end{array} \right\}$$

Another useful property satisfied by all common logics, and that we will need for the completeness result is the following:

Definition 2.37 (Independent $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$). We say that $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent, if for any finite set of keys S, and any finite set of ciphers $\mathfrak{C}$ such that no key in S appears in any element of $\mathfrak{C}$, given any key-renaming function σ, there is a key renaming σ' for which $\sigma'(K) = K$ whenever $K \in S$, and for all $C \in \mathfrak{C}$, $C\sigma \equiv_{\mathbf{C}} C\sigma'$.

In other words, $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent, if for any finite set of keys S, and any finite set of ciphers $\mathfrak{C}$ such that no key in S appears in any element of $\mathfrak{C}$, it is possible to alter any key-renaming function σ such that the altered function leaves all the elements in S unchanged, whereas on $\mathfrak{C}$ it does the same thing as the original σ. We will need this property for the general completeness theorem.

2.5.3 Interpretation

The idea of the interpretation is to describe messages that are built from blocks of strings and keys via pairing and encryption. To each valid formal expression M, the interpretation assigns a random variable $\Phi(M)$ taking values in $\overline{\texttt{strings}}$. We do not give one specific interpreting function though, we will just say that a function Φ is an interpretation if it satisfies certain properties. We assume, that a function ϕ is fixed in advance, which assigns to each formal key a key-generation algorithm. If $\Phi(B) \in \overline{\texttt{strings}}$ (constant random variable) is given for blocks, then, the rest of Φ is determined the following way: First, run the key-generation algorithm assigned by ϕ for each key in *Keys*(M). Then, using the outputs of these key-generations, translate the formal expressions according to the following rules: for each key, use the output of the corresponding key-generation. For blocks, just use $\Phi(B)$. For each pair, apply $[\cdot,\cdot]$ to the interpretations of the expressions inside the formal pair. For each formal encryption, run the encryption algorithm using as key the bitstring that was output by the key generation, to encrypt the interpretation of the formal expression inside the formal encryption. The randomness of $\Phi(M)$ comes from the initial key-generation, and from running the encryption algorithm independently for each formal encryption. We define below this notion of interpretation. We motivate it with the following example:

Example 2.11. For $M = ((\{0\}_{K_{10}}, K_5), \{K_{10}\}_{K_5})$, the interpretation is $\Phi(M) : (\Omega_{\mathcal{E}} \times \Omega_{\mathcal{E}}) \times (\Omega_{\phi(K_5)} \times \Omega_{\phi(K_{10})}) \to \overline{\texttt{strings}}$, where $\Phi(M)(\omega_1, \omega_2, \omega_3, \omega_4)$ is

$$\big[[\mathcal{E}(\phi(K_{10})(\omega_4), \Phi(0))(\omega_1), \phi(K_5)(\omega_3)], \mathcal{E}(\phi(K_5)(\omega_3), \phi(K_{10})(\omega_4))(\omega_2)\big].$$

There are four instances of randomness, two coming from the generation of keys by the key-generation algorithm (for K_5 and for K_{10}), and the other two from the two encryptions $(\{0\}_{K_{10}})$ and $(\{K_{10}\}_{K_5})$.

Definition 2.38 (Interpretation of Formal Expressions). Let $\Pi = (\{\mathcal{K}_i\}_{i\in I}, \mathcal{E}, \mathcal{D}, \approx)$ be a general symmetric encryption scheme with some index set I, with $\{(\Omega_{\mathcal{K}_i}, \Pr_{\mathcal{K}_i})\}_{i\in I}$ denoting the probability fields for key generation, and with $(\Omega_{\mathcal{E}}, \Pr_{\mathcal{E}})$ denoting the probability field for the randomness of encryption. Let $\mathbf{Exp}_{\mathcal{V}}$ be a set of valid expressions. For each valid expression M, let the probability space $(\Omega_M, \Pr_M)$ be defined recursively as

$$\begin{aligned}
&(\Omega_K, \Pr{}_K) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \text{ for } K \in \mathbf{Keys};\\
&(\Omega_B, \Pr{}_B) := (\{\omega_0\}, \mathbf{1}_{\{\omega_0\}}) \text{ for } B \in \mathbf{Blocks};\\
&(\Omega_{(M,N)}, \Pr{}_{(M,N)}) := (\Omega_M \times \Omega_N, \Pr{}_M \otimes \Pr{}_N);\\
&(\Omega_{\{M\}_K}, \Pr{}_{\{M\}_K}) := (\Omega_{\mathcal{E}} \times \Omega_M, \Pr{}_{\mathcal{E}} \otimes \Pr{}_M).
\end{aligned}$$

Where $(\{\omega_0\}, \mathbf{1}_{\{\omega_0\}})$ is just the trivial probability-space with one elementary event, ω_0 only; the tensor product stands for the product probability. Suppose that a function $\phi : \mathbf{Keys} \to \{\mathcal{K}_i\}_{i\in I}$ is given assigning abstract keys to key generation algorithms, such that $\phi(K) = \phi(K')$ if and only if $K \equiv_{\mathbf{K}} K'$. Let $\iota : \{1, .., |\textit{Keys}(M)|\} \to \textit{Keys}(M)$ be a bijection enumerating the keys in *Keys*(M). Let

$$\begin{aligned}
&(\Omega_{\textit{Keys}(M)}, \Pr{}_{\textit{Keys}(M)}) :=\\
&\qquad \big(\, \Omega_{\phi(\iota(1))} \times ... \times \Omega_{\phi(\iota(|\textit{Keys}(M)|))}, \Pr{}_{\phi(\iota(1))} \otimes \cdots \otimes \Pr{}_{\phi(\iota(|\textit{Keys}(M)|))} \,\big).
\end{aligned}$$

The function $(M, M') \mapsto (\Phi_M(M') : \Omega_{M'} \times \Omega_{\mathit{Keys}(M)} \to \overline{\mathbf{strings}})$ defined whenever $M' \sqsubseteq M$, is called *an interpretation function*, if it satisfies the following properties:

$\Phi_M(B)(\omega_0, \omega) = \Phi_N(B)(\omega_0, \omega')$ for all M, N valid expressions, $B \in$ **Blocks**, $B \sqsubseteq M$, $B \sqsubseteq N$, and arbitrary $\omega \in \Omega_{\mathit{Keys}(M)}$, $\omega' \in \Omega_{\mathit{Keys}(N)}$. Let $\Phi(B) := \Phi_M(B)$.

$\Phi_M(K)(\omega_0, (\omega_1, ..., \omega_{|\mathit{Keys}(M)|})) = \phi(K)(\omega_{\iota^{-1}(K)})$ for $K \in \mathit{Keys}(M)$, with $\omega_j \in \Omega_{\phi(\iota(j))}$.

$\Phi_M((M', M''))((\omega', \omega''), \omega) = [\Phi_M(M')(\omega', \omega), \Phi_M(M'')(\omega'', \omega)]$ for all $\omega' \in \Omega_{M'}$, $\omega'' \in \Omega_{M''}$, and $\omega \in \Omega_{\mathit{Keys}(M)}$ if $(M', M'') \sqsubseteq M$.

$\Phi_M(\{M'\}_K)((\omega_{\mathcal{E}}, \omega'), \omega) = \mathcal{E}(\Phi_M(K)(\omega_0, \omega), \Phi_M(M')(\omega', \omega))(\omega_{\mathcal{E}})$ for all $\omega_{\mathcal{E}} \in \Omega_{\mathcal{E}}$, $\omega' \in \Omega_{M'}$, $\omega \in \Omega_{\mathit{Keys}(M)}$ if $\{M'\}_K \sqsubseteq M$.

Let $\Phi(M) := \Phi_M(M)$, and let $[\![M]\!]_\Phi$ denote the distribution of $\Phi(M)$.

2.5.4 Soundness

An interpretation assigns a random variable $\Phi(M)$ (and the distribution $[\![M]\!]_\Phi$ of $\Phi(M)$) to a formal valid expression M. On the set of valid expressions the equivalence $\cong$ equates expressions that a formal adversary supposedly cannot distinguish, whereas the equivalence $\approx$ equates random variables (and distributions) that a probabilistic adversary is not supposed to be able to distinguish. The question is, how the formal and the probabilistic equivalence are related through the interpretation. We say that soundness holds if $M \cong N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$, whereas we say that completeness holds if $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ implies $M \cong N$.

The key to a soundness theorem is to have enough boxes in the definition of formal equivalence, *i.e.*, there should be enough elements in $\mathcal{Q}_{\mathbf{Enc}}$. It is clear that in the extreme case, when the equivalence on encryption terms, $\equiv_{\mathbf{C}}$, is defined so that two encryption terms are equivalent iff they are the same, then soundness holds trivially for all interpretations; but this would be completely impractical, it would assume a formal adversary that can see everything inside every encryption. It is also immediate, that if soundness holds with a given $\equiv_{\mathbf{C}}$ (and a given interpretation), and $\equiv'_{\mathbf{C}}$ is such that for any to encryption terms M and N, $M \equiv'_{\mathbf{C}} N$ implies $M \equiv_{\mathbf{C}} N$ (i.e. $\equiv'_{\mathbf{C}}$ has more boxes), then, keeping the same interpretation, soundness holds with the new $\equiv'_{\mathbf{C}}$ as well. Hence, in a concrete situation, the aim is to introduce enough boxes to achieve soundness, but not too many, to sustain practicality. One way to avoid having too many boxes is to require completeness: we will see later, that obtaining completeness requires that we do not have too many boxes.

The following theorem claims the equivalence of two conditions. It is almost trivial that condition (i) implies condition (ii). The claim that (ii) implies (i) can be summarised the following way: if soundness holds for pairs of valid expressions M and M' with a special relation between them (described in (ii)), then soundness holds for all expressions (provided that they do not have encryption cycles). In other words, if $M \cong M'$ implies $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$ for certain specified pairs M and M', then $M \cong N$ implies $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ for any two pairs of valid expressions M and N.

For the definition of $\mathfrak{R}(\mathfrak{C}, S)$, see Section 2.5.2.

Theorem 2.14. *Let $\Delta = (\boldsymbol{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be a formal logic for symmetric encryption such $\equiv_{\mathbf{C}}$ is proper and for each $M \in \boldsymbol{Exp}_{\mathcal{V}}$, B-Keys$(M)$ is not cyclic in M. Let $\Pi = (\{\mathcal{K}_i\}_{i\in I}, \mathcal{E}, \mathcal{D}, \approx)$ be a general encryption scheme, Φ an interpretation of $\boldsymbol{Exp}_{\mathcal{V}}$ in Π. Then the following conditions are equivalent:*

(i) *Soundness holds for Φ: $M \cong N$, implies $\Phi(M) \approx \Phi(N)$.*

(ii) *For any $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ set of valid encryption terms, and S finite set of keys with $L_i \notin S$ $(i \in \{1, ..., n\})$, there is an element $\{C_\nu\}_{\nu\in\mu(\mathfrak{C})}$ of $\mathfrak{R}(\mathfrak{C}, S)$ such that the followings hold: if $\big\{\{N_{i_j}\}_K\big\}_{j=1}^{l} \subset \mathfrak{C}$ and $M \in \boldsymbol{Exp}_{\mathcal{V}}$ are such that*

1. *$\{N_{i_1}\}_K, \{N_{i_2}\}_K, ..., \{N_{i_l}\}_K \sqsubseteq M$,*
2. *R-Keys$(M) \subseteq S$, and*
3. *K does not occur anywhere else in M,*
4. *all visible undecryptable encryption terms in M are elements of $\mathfrak{C} \cup \{C_\nu\}_{\nu\in\mu(\mathfrak{C})}$,*

then, if we denote by M' the expression obtained by replacing in M each $\{N_{i_j}\}_K$ with $C_{\mu(\{N_{i_j}\}_K)}$, we have that $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$.

Proof. The proof of this theorem is motivated by the soundness proof in [AR02]. The idea of the proof is the following: Starting from two acyclic expressions $M_0 = M \cong N = N_0$, we create expressions $M_1, ..., M_b$ and $N_1, ..., N_{b'}$ such that M_{i+1} is obtained from M_i via a replacement of encryption terms as described in condition (ii). Acyclicity ensures that the encrypting key of the replaced encryption terms will not occur anywhere else. Similarly for N_{i+1} and N_i. We do this so that M_b and $N_{b'}$ will differ only in key renaming. Then, by condition (ii), $[\![M_{i+1}]\!]_\Phi \approx [\![M_i]\!]_\Phi$, and $[\![N_{i+1}]\!]_\Phi \approx [\![N_i]\!]_\Phi$. But, $[\![M_b]\!]_\Phi = [\![N_{b'}]\!]_\Phi$, and therefore the theorem follows.

Now in more detail. Condition (ii) follows from (i) easily: For any set $\{C_{\mu(\{N_{i_j}\}_K)}\}_{i=1}^l$ provided by Proposition 2.13, the encrypting key of $C_{\mu(\{N_{i_j}\}_K)}$ is not contained in S hence it is not recoverable key of M. Therefore, while computing the pattern of M', $C_{\mu(\{N_{i_j}\}_K)}$ will be replaced by the box $\Box_{\mu(\{N_{i_j}\}_K)}$, which is the same box as the one that replaces $\{N_{i_j}\}_K$ in M when the pattern of M is computed. Hence $M \cong M'$, and therefore, since soundness is assumed, and *B-Keys*(M') is not cyclic in M', we have

$$[\![M]\!]_\Phi \approx [\![M']\!]_\Phi.$$

In order to prove that (i) follows from (ii), consider two equivalent valid expressions M and N such that $M \cong N$. Then, by definition, there exists a bijection σ on **Keys** (preserving $\equiv_{\mathbf{K}}$ such that *pattern*$(M) =$ *pattern*$(N\sigma)$. This means that the "boxes" occurring in *pattern*(M) must occur in *pattern*$(N\sigma)$ and vice-versa. Also, the subexpressions of *pattern*(M) and of *pattern*$(N\sigma)$ outside the boxes must agree as well. Hence,

$$\textit{R-Keys}(M) = \textit{R-Keys}(N\sigma) = \textit{R-Keys}(N)\sigma.$$

Let $L_1, L_2, \ldots, L_b$ $(L_i \neq L_j$ if $i \neq j)$ denote the keys in *B-Keys*(M), and let $L_1', L_2', \ldots, L_{b'}'$ $(L_i' \neq L_j'$ if $i \neq j)$ denote the keys in *B-Keys*$(N)\sigma$. *B-Keys*(M) and *B-Keys*(N) (and therefore

B-Keys$(N\sigma)$ as well) are not cyclic by hypothesis, so without loss of generality, we can assume that the L_i's and the L_i''s are numbered in such a way that L_i encrypts L_j (and L_i' encrypts L_j') only if $i < j$ (for a more detailed argument about this, see [AR02]; intuitively this means that those keys in *B-Keys*(M) that are deeper in M have a higher number).

Consider now the set of expressions that are subexpressions of M or N and have the form $\{M'\}_{L_i}$ or $\{N'\}_{L_i'}$, and also, the set S. Condition (ii) then provides the set with elements of the form $C_{\mu(\{M'\}_{L_i})}$ and $C_{\mu(\{N'\}_{L_i'})}$.

Let $M_0 = M$. Let M_1 be the expression obtained from M_0 by replacing all subexpressions in M_0 of the form $\{M'\}_{L_1}$ by $C_{\mu(\{M'\}_{L_1})}$ given by the assumption. Let then M_i, $i \geq 2$, be the expression obtained from M_{i-1} by replacing all subexpressions in M_{i-1} of the form $\{M'\}_{L_i}$ by $C_{\mu(\{M'\}_{L_i})}$. We do this for all $i \leq b$ and it is easy to see that in M_b replacing the subexpressions of the form $C_{\mu(\{M'\}_{L_i})}$ by $\Box_{\mu(\{M'\}_{L_i})}$ for all i, we arrive at *pattern*(M).

Note that in M_{i-1}, L_i can only occur as an encrypting key. The reason for this is that if L_i is a subexpression of M, then it has to be encrypted with some non-recoverable key, otherwise L_i would be recoverable; moreover, it has to be encrypted with some key in *B-Keys*(M) because a subexpression of M is either recoverable or ends up in a box when we construct *pattern*(M). Now, the element in *B-Keys*(M) that encrypts L_i has to be an L_j with $j < i$. But, all subexpressions in M of the form $\{M'\}_{L_j}$ were already replaced by $C_{\mu(\{M'\}_{L_j})}$ when we constructed M_j. According to the properties listed in proposition 2.13, L_i may only appear in $C_{\mu(\{M'\}_{L_j})}$ as the encrypting key, and then $L_i = L_j$, a contradiction. So L_i cannot appear in M_{i-1} in any other place than an encrypting key. Observe as well, that *R-Keys*$(M_i) =$ *R-Keys*(M).

From assumption (ii), it follows then that $[\![M_{i-1}]\!]_\Phi \approx [\![M_i]\!]_\Phi$, for all i, $1 \leq i \leq b$. Hence,

$$[\![M]\!]_\Phi = [\![M_0]\!]_\Phi \approx [\![M_b]\!]_\Phi. \tag{2.3}$$

Carrying out the same process for $N\sigma$ through $(N\sigma)_0$, $(N\sigma)_1$, ..., $(N\sigma)_{b'}$ we arrive at

$$[\![(N\sigma)]\!]_\Phi = [\![(N\sigma)_0]\!]_\Phi \approx [\![(N\sigma)_{b'}]\!]_\Phi. \tag{2.4}$$

Since we supposed that $M \cong N$, that is, *pattern*$(M) =$ *pattern*$(N\sigma)$, and therefore $M_b =$ *pattern*(M) and $(N\sigma)_{b'} =$ *pattern*$(N\sigma)$, we have

$$[\![M_b]\!]_\Phi = [\![(N\sigma)_{b'}]\!]_\Phi. \tag{2.5}$$

Then, it is clearly true that

$$[\![N]\!]_\Phi = [\![N\sigma]\!]_\Phi \tag{2.6}$$

because permuting the keys in N does not have any effect in the distributions. Putting together Equations (2.3), (2.4), (2.5) and (2.6) the soundness result follows:

$$[\![M]\!]_\Phi \approx [\![N]\!]_\Phi.$$

$\Box$

Remark 3. The reader might ask why we do not have a similar general theorem for key-cycles and KDM-like security. The reason is that this general soundness theorem tells us in which conditions the several steps of the Abadi-Rogaway hybrid argument can be carried out. One of the conditions is that by doing one step of replacement, we must obtain equivalent interpretations, provided that we have the appropriate security notion. However, in our theorem using KDM security to solve the key-cycles issue, there is only one step of replacement! All the replacements of undecryptable terms is done at once. Therefore, in a general theorem (without assuming a specific security level), the condition of the theorem would have to be exactly what we would want to prove.

Example 2.12 (Type-1 Soundness). The soundness theorem we presented earlier for type-1 encryption schemes is a special case of the theorem above. In this case $\mathbf{Exp}_{\mathcal{V}} = \mathbf{Exp}$; the equivalence relation $\equiv_{\mathbf{C}}$ is as in Example 2.6, which is proper as we mentioned in Example 2.10; and the equivalence relation $\equiv_{\mathbf{K}}$ is trivial here, all keys are equivalent. The elements $\mu \in \mathcal{Q}_{\mathbf{Enc}}$ are in one-to-one correspondence with the possible length, so the patterns that we obtain this way are essentially the same what we defined in Section 2.3.1, and the equivalence of expressions will be $\cong_1$ that we also defined there. In order to see that condition (ii) of the general soundness theorem is satisfied for type-1, we will use the following equivalent definition of type-1 secure encryption schemes: we can also say that an encryption-scheme is *type-1 secure* if no PPT adversary A can distinguish the pair of oracles $(\mathcal{E}(k,\cdot,\cdot,0), \mathcal{E}(k',\cdot,\cdot,0))$ and $(\mathcal{E}(k,\cdot,\cdot,1), \mathcal{E}(k,\cdot,\cdot,1))$ as k and k' are independently generated, that is, for all PPT adversaries A:

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot,\cdot,0),\mathcal{E}(k',\cdot,\cdot,0)}(1^\eta) = 1\right] - \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathcal{E}(k,\cdot,\cdot,1),\mathcal{E}(k,\cdot,\cdot,1)}(1^\eta) = 1\right] \leq \operatorname{neg}(\eta)$$

where the oracle $\mathcal{E}(k,\cdot,\cdot,0)$, upon the submission of two messages with equal lengths encrypts the first, and the oracle $\mathcal{E}(k,\cdot,\cdot,1)$ encrypts the second.

To show that condition (ii) of Theorem 2.14 holds, we first have to choose $\{C_\nu\}_{\nu \in \mu(\mathfrak{C})}$ for a given set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$. We can choose any family $\{C_\nu\}_{\nu \in \mu(\mathfrak{C})}$ such that all the C_ν are encrypted with the same key, let's call it L_0, that is not present in any of the $\{N_i\}_{L_i}$ (neither in M). This is possible, because, as it is easy to check, $\nu_{\text{key}} = \mathbf{Keys}$ for all $\nu \in \mathcal{Q}_{\mathbf{Enc}}$. Then, let M be as in condition (ii). We need to show that if $\{\{N_{i_j}\}_L\}_{j=1}^l \subseteq \mathfrak{C}$ and if we denote by M' the expression obtained from M by replacing each $\{N_{i_j}\}_L$ with $C_{\mu(\{N_{i_j}\}_L)}$, then $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$.

Suppose that $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$, which means that there is an adversary A that is able to distinguish the two distributions, that is

$$\Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1] - \Pr[x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

is a non-negligible function of η. We will show that this contradicts type-1 security. To this end, we construct an adversary that can distinguish between the two pair of oracles above. This adversary is the following probabilistic algorithm that access to the oracles f and g:

```
algorithm $\mathsf{B}^{f,g}(1^\eta, M)$
    for $K \in Keys(M) \setminus \{L, L_0\}$ do $\tau(K) \longleftarrow \mathcal{K}(1^\eta)$
    $y \longleftarrow \mathrm{CONVERT2}(M)$
    $b \longleftarrow \mathsf{A}(1^\eta, y)$
    **return** $b$

**algorithm** CONVERT2$(N)$
    **if** $N = K$ where $K \in$ **Keys then**
        **return** $\tau(K)$
    **if** $N = B$ where $B \in$ **Blocks then**
        **return** $B$
    **if** $N = (M_1, M_2)$ **then**
        $x \longleftarrow \mathrm{CONVERT2}(M_1)$
        $y \longleftarrow \mathrm{CONVERT2}(M_2)$
        **return** $[x, y]$
    **if** $N = \{M_1\}_L$ **then**
        $x \longleftarrow \mathrm{CONVERT2}(M_1)$
        $y \longleftarrow \mathrm{CONVERT2}(M_\nu)$   (where $C_{\mu(\{M_1\}_L)} = \{M_\nu\}_{L_0}$)
        $z \longleftarrow f(x, y)$
        **return** $z$
    **if** $N = \{M_1\}_{L_0}$ **then**
        $x \longleftarrow \mathrm{CONVERT2}(M_1)$
        $y \longleftarrow g(x, x)$
        **return** $y$
    **if** $N = \{M_1\}_K$ $(K \notin \{L, L_0\})$ **then**
        $x \longleftarrow \mathrm{CONVERT2}(M_1)$
        $y \longleftarrow \mathcal{E}(\tau(K), x)$
        **return** $y$
```

Note that the algorithm CONVERT2 does almost the same as the algorithm CONVERT in Figure 2.1, except that while CONVERT carries out all the necessary encryptions, CONVERT2 makes the oracles carry out the encryptions for L and L_0. Therefore, in the case, when the pair of oracles (f, g) is $(\mathcal{E}(k,\cdot,\cdot,0), \mathcal{E}(k',\cdot,\cdot,0))$, then CONVERT2$(M)$ will be a random sample from $[\![M]\!]_{\Phi_\eta}$, whereas if the pair of oracles used is $(\mathcal{E}(k,\cdot,\cdot,1), \mathcal{E}(k,\cdot,\cdot,1))$, then CONVERT2$(M)$ will be a random sample from $[\![M']\!]_{\Phi_\eta}$. Thus,

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,\cdot,\cdot,0),\mathcal{E}(k',\cdot,\cdot,0)}(1^\eta, M) = 1\right] \;=\; \Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

and

$$\Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,\cdot,\cdot,1),\mathcal{E}(k,\cdot,\cdot,1)}(1^\eta, M) = 1\right] \;=\; \Pr[x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

But, according to our assumption, $[\![M]\!]_\Phi$ and $[\![M']\!]_\Phi$ can be distinguished, that is,

$$\Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1] - \Pr[x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

is a non-negligible function of η and so, there is an adversary $\mathsf{B}^{f,g}(1^\eta, \cdot)$ such that

$$\Pr\left[k, k' \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,\cdot,\cdot,0),\mathcal{E}(k',\cdot,\cdot,0)}(1^\eta, M) = 1\right] - \\ \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,\cdot,\cdot,1),\mathcal{E}(k,\cdot,\cdot,1)}(1^\eta, M) = 1\right]$$

is also a non-negligible function of η. This implies that our scheme cannot be type-1 secure, which contradicts the assumption. Hence, we cannot have $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$. Hence, condition (ii) of the general soundness theorem is satisfied, so soundness holds for the type-1 case.

Example 2.13 (Type-2 Soundness). The soundness theorem we presented earlier for type-2 encryption schemes is also a special case of the theorem above. In this case $\mathbf{Exp}_\mathcal{V} = \mathbf{Exp}$; the equivalence relation $\equiv_\mathbf{C}$ is as in Example 2.7, which is proper as we mentioned in Example 2.9; and the equivalence relation $\equiv_\mathbf{K}$ is trivial here, all keys are equivalent. The elements $\mu \in \mathcal{Q}_\mathbf{Enc}$ are in one-to-one correspondence with the keys, so we can say $\mathcal{Q}_\mathbf{Enc} \equiv \mathbf{Keys}$, and thus the boxes are labelled with keys. In this case Φ gives an interpretation in the computational setting. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii) of the theorem, we can take $C_{L_i} := \{0\}_{L_i}$, and then condition (ii) is satisfied, because the following proposition holds:

Proposition 2.15. *Consider an expression M, and a key $L \in Keys(M)$. Suppose that for some expressions $M_1, M_2, ..., M_l \in$ **Exp**, $\{M_1\}_L, \{M_2\}_L, ..., \{M_l\}_L \sqsubseteq M$, and assume also that L does not occur anywhere else in M. Then, denoting by M' the expression that we get from M by replacing each of $\{M_i\}_L$ that are not contained in any of M_j ($j \neq i$) by $\{0\}_L$, $[\![M]\!]_\Phi \approx [\![M']\!]_\Phi$ holds when the expressions are interpreted with a type-2 encryption scheme.*

Proof. We can assume, without loss of generality, that $\{M_i\}_L$ is a subexpression of $\{M_j\}_L$ only if $i < j$. Suppose that $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$, which means that there is an adversary A that distinguishes the two distributions, that is

$$\Pr(x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1) - \Pr(x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1)$$

is a non-negligible function of η. We will show that this contradicts type-2 security. To this end, we construct an adversary that can distinguish between the oracles $\mathcal{E}(k, \cdot)$ and $\mathcal{E}(k, 0)$. This adversary is the following probabilistic algorithm that access to the oracle f:

```
**algorithm** $\mathsf{B}^f(1^\eta, M)$
    **for** $K \in Keys(M) \setminus \{L\}$ **do** $\tau(K) \longleftarrow \mathcal{K}(1^\eta)$
    $y \longleftarrow$ CONVERT2($M$)
    $b \longleftarrow \mathsf{A}(1^\eta, y)$
    **return** $b$

**algorithm** CONVERT2($N$)
    **if** $N = K$ where $K \in$ **Keys then**
        **return** $\tau(K)$
    **if** $N = B$ where $B \in$ **Blocks then**
```

```
        **return** $B$
    **if** $N = (N_1, N_2)$ **then**
        $x \longleftarrow$ CONVERT2$(N_1)$
        $y \longleftarrow$ CONVERT2$(N_2)$
        **return** $[x, y]$
    **if** $N = \{N_1\}_L$ **then**
        $x \longleftarrow$ CONVERT2$(N_1)$
        $y \longleftarrow f(x)$
        **return** $y$
    **if** $N = \{N_1\}_K$ $(K \neq L)$ **then**
        $x \longleftarrow$ CONVERT2$(N_1)$
        $y \longleftarrow \mathcal{E}(\tau(K), x)$
        **return** $y$
```

Note that the algorithm CONVERT2 does almost the same as the algorithm CONVERT in Figure 2.1, except that while CONVERT carries out all necessary encryptions, CONVERT2 makes the oracles carry out the encryptions for L. Therefore, in the case, when the oracle f is $\mathcal{E}(k, \cdot)$, then CONVERT2(M) will be a random sample from $[\![M]\!]_{\Phi_\eta}$, whereas if the oracle used is $\mathcal{E}(k, 0)$, then CONVERT2(M) will be a random sample from $[\![M']\!]_{\Phi_\eta}$. Thus,

$$\begin{aligned}\Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,\cdot)}(1^\eta, M) = 1\right] &= \Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1] \\ &\text{and} \\ \Pr\left[k \longleftarrow \mathcal{K}(1^\eta) : \mathsf{B}^{\mathcal{E}(k,0)}(1^\eta, M) = 1\right] &= \Pr[x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]\end{aligned}$$

But, according to our assumption, $[\![M]\!]_\Phi$ and $[\![M']\!]_\Phi$ can be distinguished, that is,

$$\Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1] - \Pr[x \longleftarrow [\![M']\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

is a non-negligible function of η and so, there is an adversary $\mathsf{B}^f(1^\eta, \cdot)$ that can distinguish the oracles $\mathcal{E}(k, \cdot)$ and $\mathcal{E}(k, 0)$, for randomly generated keys k. This implies that our scheme cannot be type-2 secure, which contradicts the assumption. Hence, we cannot have $[\![M]\!]_\Phi \not\approx [\![M']\!]_\Phi$. $\square$

Hence, condition (ii) of the general soundness theorem is satisfied, so soundness holds for the type-2 case.

Example 2.14 (Soundness for One-Time Pad). In order to see that the formal treatment of Section sec:OTP is a special case of the general formalism, take $\equiv_{\mathbf{C}}$ so that two encryption terms are equivalent, iff (again) the encryption terms have the same encrypting key. The equivalence of keys, $\equiv_{\mathbf{K}}$ is defined with the help of a length-function l on the keys: two keys are equivalent iff they have the same length. The boxes will again be indexed by the encrypting keys. Then for a set $\mathfrak{C} = \{\{N_i\}_{L_i}\}_{i=1}^n$ as in condition (ii), take $C_{L_i} := \{0_{l(L_i)-3}\}_{L_i}$ (where $0_{l(L_i)-3}$ means $l(L_i) - 3$ many 0's). It is not hard to check that within this setting, condition (ii) of the soundness theorem is satisfied, which is an immediate consequence of the following proposition:

Proposition 2.16. *Consider a valid expression $M \in$ **Exp**$_{\mathrm{OTP}}$, and a key $K_0 \in$ Keys(M). Suppose that for some expression M_0, $\{M_0\}_{K_0}$ is a subexpression of M, and assume also that K_0 does not occur anywhere else in M. Then, denoting by M' the expression that we get from M by replacing $\{M_0\}_{K_0}$ with $\{0_{l(K_0)-3}\}_{K_0}$ (where $0_{l(K_0)-3}$ denotes as string consisting of $l(K_0)-3$ many 0's), the following is true when Φ is the interpretation for OTP:*

$$[\![M]\!]_\Phi = [\![M']\!]_\Phi. \tag{2.7}$$

Proof. The basic properties of the OTP ensure that $\Phi(\{M_0\}_{K_0})$ is evenly distributed over the set of $l(K_0)$ long strings ending with 110, no matter what M_0 is. So the distribution of $\Phi(\{M_0\}_{K_0})$ agrees with the distribution of $\Phi(\{0_{l(K_0)-3}\}_{K_0})$. Also, since K_0 is assumed not to occur anywhere else, $\Phi_M(K_0)$ is independent of the interpretation of the rest of the expression M, and therefore, $\Phi(\{M_0\}_{K_0})$ and $\Phi(\{0_{l(K_0)-3}\}_{K_0})$ are both independent of the interpretation of the rest of the expression. Hence, replacing $\Phi(\{M_0\}_{K_0})$ with $\Phi(\{0_{l(K_0)-3}\}_{K_0})$ will not effect the distribution. □

2.5.5 Parsing Process

The technique that we present in this chapter will be very useful in the course of proving our completeness results. The idea can be summarised as follows: Given a sample element $x \longleftarrow [\![M]\!]_\Phi$, x is built from blocks and randomly generated keys which are paired and encrypted. Some of the keys that were used for encryption when x was built might be explicitly contained in x, and in this case, using these keys, we can decrypt those ciphers that were encrypted with these revealed keys. The problem is though, that looking at x, it might not be possible to tell where blocks, keys, ciphers and pairs are in the string of bits, since we did not assume in general that we tag strings as we did for OTP. However, and we will exploit this fact repeatedly in our proofs, if we know that x was sampled from $[\![M]\!]_\Phi$ for a fixed, known expression M, then by looking at M, we can find in x the locations of blocks, keys, ciphers and pairs, and we can also tell from M, where the key decrypting a certain cipher is located. On the following couple of pages, we present a machinery that, using the form of an expression M, extracts from an $x \longleftarrow [\![M]\!]_\Phi$ everything that is possible via decryption and depairing, and distributes the extracted elements over a special Cartesian product of copies of $\overline{\mathbf{strings}}$.

Throughout this section, we assume that $\Delta = (\mathbf{Exp}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ and an interpretation Φ in a general symmetric encryption scheme $\Pi = (\{\mathcal{K}_i\}_{i\in I}, \mathcal{E}, \mathcal{D}, \approx)$ is given.

In this chapter we will often use the notion of *subexpression occurrence* of/in M. This means a subexpression together with its position in M. The reason for this distinction is that a subexpression can occur several times in M, and we want to distinguish these occurrences. But, to avoid cumbersome notation, we will denote the subexpression occurrence just as the subexpression itself. We start by defining the notion of 0-level subexpression occurrences of an expression M:

Definition 2.39 (Level 0 Subexpression Occurrences). For an expression M, let us call *level 0 subexpression occurrences* all those subexpression occurrences in M that are not encrypted. Let

$sub_0(M)$ denote the set of all level 0 subexpression occurrences in M. We write $N \sqsubseteq_0 M$ if N is a level 0 subexpression occurrence of N in M.

For an element $x \longleftarrow [\![M]\!]_\Phi$, the first thing to do is to extract everything that is not encrypted, which means that we have to break up all pairs in x, and replace them with mathematical pairs. This process reveals the unencrypted blocks, keys and ciphers in x (i.e., the computational or statistical realisations of the 0-level subexpression occurrences).

Definition 2.40 (Blowup Function). For each valid expression M, we define the *blowup function* $\mathcal{B}(M)$, on $\overline{\mathbf{strings}}$ inductively as follows:

$$\begin{aligned}
&\mathcal{B}(K)x := x \quad \text{for } K \text{ key}\\
&\mathcal{B}(B)x := x \quad \text{for } B \text{ block}\\
&\mathcal{B}((M_1, M_2))x := (\mathcal{B}(M_1) \oplus \mathcal{B}(M_2)) \circ [\cdot,\cdot]^{-1}(x)\\
&\mathcal{B}(\{N\}_K)x := x.
\end{aligned}$$

Where $\mathcal{B}(M_1) \oplus \mathcal{B}(M_2)$ denotes the function $(x, y) \mapsto (\mathcal{B}(M_1)x, \mathcal{B}(M_2)y)$.

The element $\mathcal{B}(M)x$ is an element of $\mathcal{T}_0(M)$, which we define inductively the following way:

Definition 2.41 (Associated 0-Tree). The *0-tree associated* to a pair of expressions N and M whenever $N \sqsubseteq_0 M$, will be denoted by $\mathcal{T}_0(N, M)$, and we define it inductively as follows:

$$\begin{aligned}
&\mathcal{T}_0(K, M) := \overline{\mathbf{strings}}\\
&\mathcal{T}_0(B, M) := \overline{\mathbf{strings}}\\
&\mathcal{T}_0((M_1, M_2), M) := \mathcal{T}_0(M_1, M) \times \mathcal{T}_0(M_2, M)\\
&\mathcal{T}_0(\{M'\}_K, M) := \overline{\mathbf{strings}}
\end{aligned}$$

Let $\mathcal{T}_0(M) := \mathcal{T}_0(M, M)$.

We remind the reader that we do not identify $(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}) \times \overline{\mathbf{strings}}$ with $\overline{\mathbf{strings}} \times (\overline{\mathbf{strings}} \times \overline{\mathbf{strings}})$.

Note also that for expressions $N \sqsubseteq_0 M'$ and $N \sqsubseteq_0 M$, we have that $\mathcal{T}_0(N, M') = \mathcal{T}_0(N, M)$. Nevertheless, we included M in the definition of $\mathcal{T}_0$ since for higher order trees, which we shall define later, the M in the second argument will make a difference.

Example 2.15. For the expression

$$M = \Bigg(\Big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\Big), \Big(\Big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\Big), \{K_5\}_{K_2}\Big)\Bigg),$$

$$sub_0(M) =$$

$$= \left\{ \begin{array}{l} \{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}, K_2, \big\{\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}, \{K_5\}_{K_2}, \Big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\Big), \\ \Big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\Big), \Big(\Big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\Big), \{K_5\}_{K_2}\Big), M \end{array} \right\},$$

and

$$\mathcal{T}_0(M) = \Big(\overline{\textbf{strings}} \times \overline{\textbf{strings}}\Big) \times \Big(\big(\overline{\textbf{strings}} \times \overline{\textbf{strings}}\big) \times \overline{\textbf{strings}}\Big).$$

Blocks, keys and ciphers are replaced by $\overline{\textbf{strings}}$, pairs are replaced by $\times$. An element x sampled from $[\![M]\!]_\Phi$ looks like

$$\Big[\, [\, c_1 \, , \, c_2 \,] \, , \, \Big[\, [\, k \, , \, c_3 \,] \, , \, c_4 \Big] \Big]$$

where c_1 is a sample from $[\![\{0\}_{K_6}]\!]_\Phi$, c_2 is a sample from $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$, k is a sample from $[\![K_2]\!]_\Phi$, c_3 is a sample from $[\![\big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}]\!]_\Phi$, and c_4 is a sample from $[\![\{K_5\}_{K_2}]\!]_\Phi$. When we apply the blow-up function to this element x, we obtain

$$\Big(\, (\, c_1 \, , \, c_2 \,) \, , \, \Big(\, (\, k \, , \, c_3 \,) \, , \, c_4 \Big) \Big)$$

which is an element of $\mathcal{T}_0(M)$.

Proposition 2.17. *For an expression M, if $x \longleftarrow [\![M]\!]_\Phi$, then $\mathcal{B}(M)(x) \in \mathcal{T}_0(M)$.*

Proof. Immediate from the definitions of $\mathcal{B}$ and $\mathcal{T}_0$. □

Perhaps it is even clearer if we label the copies of $\overline{\textbf{strings}}$ in $\mathcal{T}_0(M)$ with the formal expressions that they belong to:

$$\begin{aligned}
&\mathcal{T}_0'(K, M) := \overline{\textbf{strings}}_K\\
&\mathcal{T}_0'(B, M) := \overline{\textbf{strings}}_B\\
&\mathcal{T}_0'((M_1, M_2), M) := \mathcal{T}_0'(M_1, M) \times \mathcal{T}_0'(M_2, M)\\
&\mathcal{T}_0'(\{M'\}_K, M) := \overline{\textbf{strings}}_{\{M'\}_K}.
\end{aligned}$$

In our example,

$$\mathcal{T}_0'(M, M) = \Big(\mathrm{s}_{\{0\}_{K_6}} \times \mathrm{s}_{\{\{K_7\}_{K_1}\}_{K_4}}\Big) \times \Big(\big(\mathrm{s}_{K_2} \times \mathrm{s}_{\{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}}\big) \times \mathrm{s}_{\{K_5\}_{K_2}}\Big),$$

where we used s as a shorthand for $\overline{\textbf{strings}}$.

In the previous example, c_4 is a random sample from $[\![\{K_5\}_{K_2}]\!]_\Phi$, and the function that projects onto the last copy of $\overline{\textbf{strings}}$ in $\mathcal{T}_0(M)$, namely, onto $\overline{\textbf{strings}}_{\{K_5\}_{K_2}}$, extracts c_4 from the blow-up. Similarly, projecting onto the other copies of $\overline{\textbf{strings}}$, we extract samples form $[\![\{0\}_{K_6}]\!]_\Phi$, $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$ etc. To implement this idea in the general situation, we define what we can call the "0-Get Function" $\mathcal{G}_0(N, M)$ for an expression M and a subexpression occurrence N, whenever N is not encrypted in M. For $x \longleftarrow [\![M]\!]_\Phi$, the purpose of $\mathcal{G}_0(N, M)$ is to extract from $\mathcal{B}(M)x$ the sample of $[\![N]\!]_\Phi$ that was used for computing x. The precise definition is the following:

Definition 2.42 (0-Get Function). For subexpression occurrences $N \sqsubseteq_0 N' \sqsubseteq_0 M$, we define the *0-get function associated* to the triple (N, N', M), $\mathcal{G}_0(N, N', M) : \mathcal{T}_0(N', M) \to \mathcal{T}_0(N, M)$ inductively in N' as follows:

$$\mathcal{G}_0(N, N, M) := \mathrm{id}_{\mathcal{T}_0(N,M)}$$

$$\mathcal{G}_0(N, (M_1, M_2), M) := \begin{cases} \mathcal{G}_0(N, M_1, M) \circ \pi^1_{\mathcal{T}_0(M_1,M)\times\mathcal{T}_0(M_2,M)} & \text{if } N \text{ occurs in } M_1, \\ \mathcal{G}_0(N, M_2, M) \circ \pi^2_{\mathcal{T}_0(M_1,M)\times\mathcal{T}_0(M_2,M)} & \text{otherwise} \end{cases}$$

We define $\mathcal{G}_0(N, M) := \mathcal{G}_0(N, M, M)$.

Example 2.16. In the previous example,

$$\mathcal{G}_0\big(\{0\}_{K_6}, M\big),\ \ \mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big) : \mathcal{T}_0(M) \to \overline{\mathbf{strings}}$$

$$\mathcal{G}_0\big(\{0\}_{K_6}, M\big)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = x_1,$$

$$\mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = x_2,$$

etc; that is, $\mathcal{G}_0\big(\{0\}_{K_6}, M\big)$ does the projection onto $\overline{\mathbf{strings}}_{\{0\}_{K_6}}$, $\mathcal{G}_0\big(\{\{K_7\}_{K_1}\}_{K_4}, M\big)$ does the projection onto $\overline{\mathbf{strings}}_{\{\{K_7\}_{K_1}\}_{K_4}}$, etc.

Observe, that for two expressions M and N, if $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, then for any $M' \in sub_0(M)$, there is a unique $N' \in sub_0(N)$ such that $\mathcal{G}_0(M', M) = \mathcal{G}_0(N', N)$. This motivates the following definition:

Definition 2.43 (Same Position of Subexpression Occurrences). For two expressions M and N, if $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, we say that $M' \in sub_0(M)$ and $N' \in sub_0(M)$ *are in the same position at level 0*, if

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(N', N).$$

Let

$$\Gamma_0(N, M) : sub_0(M) \to sub_0(N)$$

denote the unique bijection such that

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(\Gamma_0(N, M)M', N)$$

for all $M' \in sub_0(M)$.

Example 2.17. Let $N = ((0, 0), ((0, 0), 0)$. Then, if M denotes the expression from the previous examples, $\mathcal{T}_0(N) = \mathcal{T}_0(M)$. Enumerating the 0's in N, we get the subexpression occurrences $0_1 = 0$, $0_2 = 0$, $0_3 = 0$, $0_4 = 0$ and $0_5 = 0$, with $N = ((0_1, 0_2), ((0_3, 0_4), 0_5)$. We have that:

$$\begin{aligned}
&\Gamma_0(N, M)\{0\}_{K_6} = 0_1 \\
&\Gamma_0(N, M)\{\{K_7\}_{K_1}\}_{K_4} = 0_2 \\
&\Gamma_0(N, M)K_2 = 0_3 \\
&\Gamma_0(N, M)\big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5} = 0_4 \\
&\Gamma_0(N, M)\{K_5\}_{K_2} = 0_5 \\
&\Gamma_0(N, M)\Big(\{0\}_{K_6}, (\{\{K_7\}_{K_1}\}_{K_4}\Big) = (0_1, 0_2)
\end{aligned}$$

etc.

For an expression M, let $\mathcal{C}_M$ denote the set of all those subexpression occurrences in M which are ciphers encrypted by recoverable keys, i.e.,

$$\mathcal{C}_M = \{\{M'\}_K \sqsubseteq M \mid \{M'\}_K \in \mathit{vis}\,(M) \,\text{and}\, K \in \textit{R-Keys}(M)\}.$$

We emphasise that in the previous definition we are referring to subexpression occurrences, that is, if an encryption term is encrypted with a recoverable key occurs twice in M, then it will be listed twice in $\mathcal{C}_M$. Since we assume that the elements of this set are encrypted by recoverable keys, it is possible to decrypt these elements one after the other, using only information containing M. Therefore, it is possible to enumerate the elements of this set in an order in which we can decrypt them by taking keys from M, decrypting what is possible with these keys and hence revealing more keys and then decrypting again with those keys etc. Let the total number of this set be denoted by $c(M)$. Then

$$\mathcal{C}_M = \{C^1, C^2, ..., C^{c(M)}\}.$$

Note that this enumeration is not unique. Also, note that the numbering does not mean that you can decrypt the ciphers only in this order. Let C^i_{key} denote the key that is used in the encryption C^i and let C^i_{text} denote the encrypted expression.

Example 2.18. In our example, the only possible way to enumerate is

$$\begin{aligned}
C^1 &= \{K_5\}_{K_2}\\
C^2 &= \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\\
C^3 &= \{K_6\}_{K_5}\\
C^4 &= \{0\}_{K_6}.
\end{aligned}$$

Now, to each expression M, we associate the "1-Decrypting Function" $\mathcal{D}_1(M)$. It acts on $\mathcal{T}_0(M)$ and works as follows: for any $t \in \mathcal{T}_0(M)$, the function $\mathcal{D}_1(M)$ extracts $\mathcal{G}_0(C^1, M)t$ from $\overline{\mathbf{strings}}_{C^1}$, $\mathcal{G}_0(C^1_{\text{key}}, M)t$ from $\overline{\mathbf{strings}}_{C^1_{\text{key}}}$, and with the latter decrypts the former if that is possible (namely, if they are of the right form: the former a cipher and the latter a key). The result is then broken into mathematical pairs, and what we get this way is put in the last component of the set $\overline{\mathbf{strings}} \times \{0\} \times \mathcal{T}_0(C^1_{\text{text}})$, while $\mathcal{G}_0(C^1_{\text{key}}, M)t$ goes into the first component. That is, the following element is created:

$$\Bigg(\ \mathcal{G}_0(C^1_{\text{key}}, M)t\ ,\ 0\ ,\ \mathcal{B}(C^i_{\text{text}})\Big(\mathcal{D}\big(\mathcal{G}_0(C^1_{\text{key}}, M)t, \mathcal{G}_0(C^1, M)t\big)\Big)\ \Bigg).$$

If $\big(\mathcal{G}_0(C^1_{\text{key}}, M)t, \mathcal{G}_0(C^1, M)t\big) \notin \mathrm{Dom}_{\mathcal{D}}$, then $\mathcal{D}_1(M)$ outputs $(0, 0, 0)$. The rest of $\mathcal{T}_0(M)$ is left untouched. We warn the reader for the similarity of notations between the decryption algorithm of the encryption scheme $\mathcal{D}(\cdot, \cdot)$, and the 1-Decrypting function $\mathcal{D}_1(\cdot)$. This notation is convenient as $\mathcal{D}_i(M)$ is the function that decrypts the ciphers encrypted with recoverable keys at level-i. We will always index this functions with the respective index i to avoid confusions.

Let us introduce the notation

$$\mathcal{T}_0^{C^1}(M) = \left\{t \in \mathcal{T}_0(M) \ \middle|\ \big(\mathcal{G}_0(C^1_{\text{key}}, M)t, \mathcal{G}_0(C^1, M)t\big) \in \mathrm{Dom}_{\mathcal{D}}\right\}.$$

Then,

Definition 2.44 (1-Decrypting Function). For expressions $N \sqsubseteq_0 M$, we define the function $\mathcal{D}_1(N,M)$ on $\mathcal{T}_0(M)$ inductively as follows: Let $t \in \mathcal{T}_0(M)$. Then

$$\begin{aligned}
&\mathcal{D}_1(K,M)t := \mathcal{G}_0(K,M)t\\
&\mathcal{D}_1(B,M)t := \mathcal{G}_0(B,M)t\\
&\mathcal{D}_1(\{M'\}_K,M)t := \mathcal{G}_0(\{M'\}_K,M)t \quad \text{if } K \notin \textit{R-Keys}(M)\\
&\mathcal{D}_1((M_1,M_2),M)t := \big(\mathcal{D}_1(M_1,M)t\ ,\ \mathcal{D}_1(M_2,M)t\big)\\
&\mathcal{D}_1(C^j,M)t :=
\end{aligned}$$

$$:= \begin{cases} \Big(\mathcal{G}_0(C^1_{\text{key}},M)t, 0, \mathcal{B}(C^1_{\text{text}})\big(\mathcal{D}(\mathcal{G}_0(C^1_{\text{key}},M)t, \mathcal{G}_0(C^1,M)t)\big)\Big) & \text{if } t \in \mathcal{T}_0^{C^1}(M) \text{ and } j=1\\ (0,0,0) \qquad \text{if } t \notin \mathcal{T}_0^{C^1}(M) \ \text{ and } \ j=1 & \\ \mathcal{G}_0(C^j,M)t \qquad \text{if } j>1 & \end{cases}$$

We introduce the notation $\mathcal{D}_1(M) := \mathcal{D}_1(M,M)$, this is what we will be interested in.

We remark, that it is not important how we define $\mathcal{D}_1(C^1,M)t$ when $t \notin \mathcal{T}_0^{C^1}(M)$, we will not need that. We chose $(0,0,0)$ just for convenience.

Example 2.19. In our running example we have

$$M = \bigg(\Big(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\Big), \Big(\Big(K_2, \big\{(\{001\}_{K_3}, \{K_6\}_{K_5})\big\}_{K_5}\Big), \{K_5\}_{K_2}\Big)\bigg).$$

With the choice $C^1 = \{K^5\}_{K_2}$, we obtain

$$\mathcal{D}_1(M)\big((x_1,x_2),((x_3,x_4),x_5)\big) = \begin{cases} \Big((x_1,x_2), \Big((x_3,x_4), \big(x_3, 0, \mathcal{B}(\{K_5\}_{K_2})(\mathcal{D}(x_3,x_5))\big)\Big)\Big) & \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{if } (x_3,x_5) \in \mathrm{Dom}_{\mathcal{D}} & \\ \big((x_1,x_2),((x_3,x_4),(0,0,0)\big) \quad \text{otherwise} & \end{cases}$$

The target set of $\mathcal{D}_0(M)$ is naturally not $\mathcal{T}_0(M)$, because instead of the copy of $\overline{\mathbf{strings}}$ corresponding to C^1 we now have a set of the form $\overline{\mathbf{strings}} \times 0 \times \mathcal{T}_0(C^1_{\text{text}})$. We will call this new set $\mathcal{T}_1(M)$, and so we extend the definition of $\mathcal{T}_0$ to higher order, up to $\mathcal{T}_{c(M)}(M)$. First we need the following:

Definition 2.45 (Level i Subexpression Occurrences). We will say that a subexpression occurrence $N \sqsubseteq M$ is level i with respect to $\mathcal{C}_M$, and denote this relation by $N \sqsubseteq_i M$, if the occurrence N is not in the occurrence C^j whenever $i<j$. Let *sub*$_i(M)$ denote the set of level i subexpression occurrences.

Notice, that the level i subexpression occurrences are all those which are revealed once C^1, C^2, ... ,C^i are decrypted.

Definition 2.46 (Associated i-Tree). We inductively define the *i-tree associated* to a pair of expressions $N \sqsubseteq_i M$, and denote it by $\mathcal{T}_i(N,M)$:

$$\begin{aligned}
&\mathcal{T}_i(K,M) ::= \overline{\mathbf{strings}}\\
&\mathcal{T}_i(B,M) := \overline{\mathbf{strings}}\\
&\mathcal{T}_i((M_1,M_2),M) := \mathcal{T}_i(M_1,M) \times \mathcal{T}_i(M_2,M)\\
&\mathcal{T}_i(C^j,M) := \begin{cases} \overline{\mathbf{strings}} \times \{0\} \times \mathcal{T}_i(C^j_{\text{text}},M) & \text{if } j \le i\\ \overline{\mathbf{strings}} & \text{otherwise}\end{cases}\\
&\mathcal{T}_{i-1}(\{M'\}_K,M) := \overline{\mathbf{strings}} \qquad \text{for } K \notin \textit{R-Keys}(M)
\end{aligned}$$

Let $\mathcal{T}_i(M) := \mathcal{T}_i(M,M)$.

Note that we only “open” the encryptions performed with the keys in *R-Keys*(M) and at each step i we only open the C^j such that $j \le i$.

Fact 2.47. For any expressions M and N, we have that $\mathcal{T}_i(M) \cap \mathcal{T}_i(N) = \emptyset$ or $\mathcal{T}_i(M) = \mathcal{T}_i(N)$.

Similarly, we need to define $\mathcal{G}_i(N,M)$ and $\mathcal{D}_i(M)$ for $0 < i \le c(M)$. The first one projects onto the copy of $\overline{\mathbf{strings}}$ in $\mathcal{T}_i(M)$ that corresponds to N, and the second maps an element in $\mathcal{T}_{i-1}(M)$ into $\mathcal{T}_i(M)$ decrypting the string corresponding to C^i with the appropriate key.

Definition 2.48 (i-Get Function). For subexpression occurrences $N \sqsubseteq_i M$, $N' \sqsubseteq_i M$ ($0 \le i \le c(M)$) such that N occurs in N', we define the map *i-get-function associated* to the triple (N,N',M), $\mathcal{G}_i(N,N',M) : \mathcal{T}_i(N',M) \to \mathcal{T}_i(N,M)$ inductively as follows:

$$\begin{aligned}
&\mathcal{G}_i(N,N,M) := \mathrm{id}_{\mathcal{T}_i(N,M)}\\
&\mathcal{G}_i(N,(M_1,M_2),M) := \begin{cases} \mathcal{G}_i(N,M_1,M) \circ \pi^1_{\mathcal{T}_i(M_1,M)\times\mathcal{T}_i(M_2,M)} & \text{if } N \sqsubseteq M_1\\ \mathcal{G}_i(N,M_2,M) \circ \pi^2_{\mathcal{T}_i(M_1,M)\times\mathcal{T}_i(M_2,M)} & \text{otherwise}\end{cases}\\
&\mathcal{G}_i(N,C^j,M) := \mathcal{G}_i(N,C^j_{\text{text}},M) \circ \pi^3_{\mathcal{T}_i(C^j_{\text{key}},M)\times\{0\}\times\mathcal{T}_i(C^j_{\text{text}},M)}, \text{ for } j \le i, N \ne C^j
\end{aligned}$$

Define

$$\mathcal{G}_i(N,M) := \mathcal{G}_i(N,M,M).$$

Definition 2.49 (Same Position of Subexpression Occurrences). For two expressions M and N, if $\mathcal{T}_i(M) = \mathcal{T}_i(N)$, we say that $M' \in \mathit{sub}_i(M)$ and $N' \in \mathit{sub}_i(M)$ *are in the same position at level* i, if

$$\mathcal{G}_i(M',M) = \mathcal{G}_i(N',N).$$

Let

$$\Gamma_i(N,M) : \mathit{sub}_i(M) \to \mathit{sub}_i(N)$$

denote the unique bijection such that

$$\mathcal{G}_i(M',M) = \mathcal{G}_i(\Gamma_i(N,M)M',N)$$

for all $M' \in \mathit{sub}_i(N)$.

Let

$$\mathcal{T}_{i-1}^{C^i}(M) = \left\{t \in \mathcal{T}_{i-1}(M) \;\middle|\; \left(\mathcal{G}_{i-1}(C^i_{\text{key}}, M)t, \mathcal{G}_{i-1}(C^i, M)t\right) \in \mathrm{Dom}_{\mathcal{D}}\right\}.$$

Definition 2.50 (i-Decrypting Function). For expressions $N \sqsubseteq_{i-1} M$ and $1 \le i \le c(M)$, we define the map $\mathcal{D}_i(N, M) : \mathcal{T}_{i-1}(M) \to \mathcal{T}_i(N, M)$ inductively as follows: Let $t \in \mathcal{T}_{i-1}(M)$

$\mathcal{D}_i(K, M)t := \mathcal{G}_{i-1}(K, M)t$
$\mathcal{D}_i(B, M)t := \mathcal{G}_{i-1}(B, M)t$
$\mathcal{D}_i(\{M'\}_K, M)t := \mathcal{G}_{i-1}(\{M'\}_K, M)t$ if $K \notin$ ***R-Keys***(M)
$\mathcal{D}_i((M_1, M_2), M)t := (\mathcal{D}_i(M_1, M)t, \mathcal{D}_i(M_2, M)t)$
$\mathcal{D}_i(C^j, M)t :=$

$$= \begin{cases} \left(\mathcal{G}_{i-1}(C^j_{\text{key}}, M)t, 0, \mathcal{D}_i(C^j_{\text{text}}, M)\right) & \text{if } j < i \\ \left(\mathcal{G}_{i-1}(C^i_{\text{key}}, M)t, 0, \mathcal{B}(C^i_{\text{text}})\big(\mathcal{D}(\mathcal{G}_{i-1}(C^i_{\text{key}}, M)t, \mathcal{G}_{i-1}(C^i, M)t)\big)\right), t \in \mathcal{T}_{i-1}^{C^i}(M), j = i \\ (0,0,0) \qquad \text{if } t \notin \mathcal{T}_{i-1}^{C^i}(M) \text{ and } j = i \\ \mathcal{G}_{i-1}(C^j, M)t \qquad \text{if } j > i \end{cases}$$

Let

$$\mathcal{D}(M) := \mathcal{D}_{c(M)}(M) \circ ... \circ \mathcal{D}_1(M) \circ \mathcal{B}(M)$$

The composition of functions $\mathcal{D}_i(M)$ (in order) decrypt all the ciphers that are encrypted with recoverable keys. At the end, $\mathcal{D}(M)$ decrypts all ciphers encrypted with recoverable keys upon an input from sampling $[\![M]\!]_\Phi$.

Example 2.20. In our on-going example,

$$M = \left(\left(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\right), \left(\left(K_2, \{(\{001\}_{K_3}, \{K_6\}_{K_5})\}_{K_5}\right), \{K_5\}_{K_2}\right)\right),$$

If y is a sample from $[\![M]\!]_\Phi$, then $\mathcal{D}(M)y$ has the form

$$\left(\left((y_6, 0, 0), y_1\right), \left(\left(y_2, (y_5, 0, (y_3, (y_5, 0, y_6)))\right), (y_2, 0, y_5)\right)\right),$$

Where y_2, y_5, y_6 are outcomes of the key-generation algorithms $\mathcal{K}_{\phi(K_2)}$, $\mathcal{K}_{\phi(K_5)}$, $\mathcal{K}_{\phi(K_6)}$ respectively, y_1 is an undecryptable sample element from $[\![\{\{K_7\}_{K_1}\}_{K_4}]\!]_\Phi$, and y_3 is an undecryptable sample from $[\![\{001\}_{K_3}]\!]_\Phi$. Moreover, $(y_6, 0, 0)$ indicates that the key y_6 encrypts the plaintext 0, $(y_2, 0, y_5)$ indicates that the key y_2 encrypts the plaintext y_5 (which is also a key), and so on.

The following lemma essentially claims that if the interpretation is such that conditions (i) and (ii) below hold, then for any two valid expressions M and N, the distribution of $\mathcal{D}(M)x$, where x is sampled from $[\![M]\!]_\Phi$ (let $\mathcal{D}(M)([\![M]\!]_\Phi)$ denote this distribution), is indistinguishable from the distribution of $\underline{\mathcal{D}(N)y}$, where y is sampled from $[\![N]\!]_\Phi$ whenever $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$.

For a function f on $\overline{\textbf{strings}}$, let $f([\![M]\!]_\Phi)$ denote the probability distribution of $f(x)$ as x is sampled from $[\![M]\!]_\Phi$.

Lemma 2.18. *Let $\Delta = (\textbf{\textit{Exp}}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be a formal logic for symmetric encryption, and let Φ be an interpretation of $\textbf{\textit{Exp}}_{\mathcal{V}}$ in $\Pi = (\{\mathcal{K}_i\}_{i\in I}, \mathcal{E}, \mathcal{D}, \approx)$. Suppose that this realisation satisfies the following properties for any $K, K', K'' \in$ **Keys**, $B \in$ **Blocks**, $M, M', N \in \textbf{\textit{Exp}}_{\mathcal{V}}$:*

(i) *no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ are equivalent with respect to $\approx$; that is, keys, blocks, pairs, ciphers are distinguishable.*

(i) *If $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$.*

Let M and N be valid formal expressions. Let $\mathcal{C}_M = \{C_M^1, ...C_M^{c(M)}\}$ be an enumeration of all ciphers encrypted by recoverable keys in M such that they can be decrypted in this order. Then, $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$ implies that $c(M) = c(N)$, and $\mathcal{C}_N = \{C_N^1, ..., C_N^{c(N)}\}$ can be enumerated in the order of decryption such that $\Gamma_{c(M)}(N, M)C_M^i = C_N^i$. Moreover, with this enumeration of $\mathcal{C}_N$, $\mathcal{D}_i(M) = \mathcal{D}_i(N)$, and

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi)$$

Proof. Let M and N be expressions such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. Since we assumed condition (i) and since the equivalence $\approx$ is assumed to be invariant under depairing, the pairs that are not encrypted in M and in N must be in the same positions, and so $\mathcal{B}(M) = \mathcal{B}(N)$ must hold. Since the blow-up function is obtained by repeated application of the inverse of the pairing function, projecting and coupling,

$$\mathcal{B}(M)([\![M]\!]_\Phi) \approx \mathcal{B}(N)([\![N]\!]_\Phi). \tag{2.8}$$

As mentioned in Proposition 2.17, if x is sampled from $[\![M]\!]_\Phi$, then $\mathcal{B}(M)x \in \mathcal{T}_0(M)$. Therefore,

$$\mathcal{T}_0(M) = \mathcal{T}_0(N).$$

Since $\mathcal{T}_0(M) = \mathcal{T}_0(N)$, there is a unique bijection

$$\Gamma_0(N, M) : \textit{sub}_0(M) \to \textit{sub}_0(N)$$

that satisfies

$$\mathcal{G}_0(M', M) = \mathcal{G}_0(\Gamma_0(N,M)M', N).$$

Let $C_M^1 = \{C_{M,\text{text}}^1\}_{C_{M,\text{key}}^1}$ and $L_1 := \Gamma_0(N,M)C_{M,\text{key}}^1$. L_1 must be a key for the following reason:

$$(\mathcal{G}_0(C_{M,\text{key}}^1, M) \circ \mathcal{B}(M))([\![M]\!]_\Phi) \approx (\mathcal{G}_0(C_{M,\text{key}}^1, M) \circ \mathcal{B}(M))([\![N]\!]_\Phi),$$

since we again apply the same function, $\mathcal{G}_0(C_{M,\text{key}}^1, M) \circ \mathcal{B}(M)$ on $[\![M]\!]_\Phi$ and $[\![N]\!]_\Phi$, and this function is made up of depairing, projecting and coupling. But, for the left hand side we clearly have

$$(\mathcal{G}_0(C_{M,\text{key}}^1, M) \circ \mathcal{B}(M))([\![M]\!]_\Phi) = [\![C_{M,\text{key}}^1]\!]_\Phi,$$

and for the right hand side,

$$(\mathcal{G}_0(C_{M,\text{key}}^1, M) \circ \mathcal{B}(M))([\![N]\!]_\Phi) = (\mathcal{G}_0(L_1, N) \circ \mathcal{B}(N))([\![N]\!]_\Phi) = [\![L_1]\!]_\Phi.$$

Therefore, by assumption (i) L_1 must be a key. Similarly,

$$(\mathcal{G}_0(C^1_M, M) \circ \mathcal{B}(M))(\llbracket M \rrbracket_\Phi) \approx (\mathcal{G}_0(C^1_M, M) \circ \mathcal{B}(M))(\llbracket N \rrbracket_\Phi).$$

The left-hand side equals $\llbracket C^1_M \rrbracket_\Phi$, hence we need to have an interpretation of a cipher on the right too, implying that for some N' expression and L key,

$$\Gamma_0(N, M) C^1_M = \{N'\}_L$$

and hence

$$\mathcal{G}_0(C^1_M, M) = \mathcal{G}_0(\{N'\}_L, N). \tag{2.9}$$

Then, according to the foregoing,

$$\big(\mathcal{G}_0(C^1_{M,\text{key}}, M), \mathcal{G}_0(C^1_M, M)\big) \circ \mathcal{B}(M)) = \big(\mathcal{G}_0(L_1, N), \mathcal{G}_0(\{N'\}_L, N)\big) \circ \mathcal{B}(N),$$

and therefore,

$$\Big(\big(\mathcal{G}_0(C^1_{M,\text{key}}, M), \mathcal{G}_0(C^1_M, M)\big) \circ \mathcal{B}(M)\Big)(\llbracket M \rrbracket_\Phi) \approx \\ \Big(\big(\mathcal{G}_0(L_1, N), \mathcal{G}_0(\{N'\}_L, N)\big) \circ \mathcal{B}(N)\Big)(\llbracket N \rrbracket_\Phi).$$

But, the left-hand side equals $\llbracket (C^1_{M,\text{key}}, C^1) \rrbracket_\Phi$, whereas the right-hand side is $\llbracket (L_1, \{N'\}_L) \rrbracket_\Phi$, so we have

$$\llbracket (C^1_{M,\text{key}}, C^1_M) \rrbracket_\Phi \approx \llbracket (L_1, \{N'\}_L) \rrbracket_\Phi.$$

By assumption (ii) then, $L = L_1$ follows, because $C^1_M = \{C^1_{M,\text{text}}\}_{C^1_{M,\text{key}}}$. But then we can choose the first element of $\mathcal{C}_N$ to be the occurrence $\{N'\}_{L_1}$, and with this choice,

$$\mathcal{D}_1(M) = \mathcal{D}_1(N).$$

Therefore

$$\mathcal{D}_1(M)(\mathcal{B}(M)(\llbracket M \rrbracket_\Phi)) \approx \mathcal{D}_1(N)(\mathcal{B}(N)(\llbracket N \rrbracket_\Phi)),$$

and therefore,

$$\mathcal{T}_1(M) = \mathcal{T}_1(N),$$

because $\mathcal{D}_1(M)(\mathcal{B}(M)(\llbracket M \rrbracket_\Phi))$ gives a distribution on $\mathcal{T}_1(M)$, and $\mathcal{D}_1(N)(\mathcal{B}(N)(\llbracket N \rrbracket_\Phi))$ gives a distribution on $\mathcal{T}_1(N)$.

An argument similar to the one above shows that

$$\mathcal{D}_2(M) = \mathcal{D}_2(N).$$

Namely, there is a unique bijection

$$\Gamma_1(N, M) : \textit{sub}_1(M) \to \textit{sub}_1(N)$$

satisfying

$$\mathcal{G}_1(M', M) = \mathcal{G}_1(\Gamma_1(N, M)M', N).$$

Then, just as we proved for L_1, $L_2 := \Gamma_1(N,M)C^2_{\text{key}}$ must be a key, and

$$\Gamma_1(N,M)C^2 = \{N''\}_{L_2}$$

for some N'' expression, implying that

$$\mathcal{D}_2(M) = \mathcal{D}_2(N).$$

And so on. So

$$\mathcal{D}_{c(M)}(M) \circ ... \circ \mathcal{D}_1(M)(\mathcal{B}(M)(\llbracket M \rrbracket_\Phi)) \approx \mathcal{D}_{c(M)}(N) \circ ... \circ \mathcal{D}_1(N)(\mathcal{B}(N)(\llbracket N \rrbracket_\Phi)),$$

since the functions applied on $\llbracket M \rrbracket_\Phi$ and $\llbracket N \rrbracket_\Phi$ are the same, and they are made up only of depairing, projecting, coupling and decrypting. Then, $c(M) \leq c(N)$. Reversing the role of M and N in the argument, we get that $c(N) \leq c(M)$, and so $c(M) = c(N)$. Hence,

$$\mathcal{D}(M) = \mathcal{D}(N),$$

and

$$\mathcal{D}(M)(\llbracket M \rrbracket_\Phi) = \mathcal{D}(N)(\llbracket N \rrbracket_\Phi).$$

□

We illustrate our proof with the following example:

Example 2.21. Suppose again, that

$$M = \left(\left(\{0\}_{K_6}, \{\{K_7\}_{K_1}\}_{K_4}\right), \left(\left(K_2, \left\{(\{001\}_{K_3}, \{K_6\}_{K_5})\right\}_{K_5}\right), \{K_5\}_{K_2}\right)\right),$$

and assume that conditions (i) and (ii) of the lemma are satisfied. Suppose that N is also a valid expression such that $\llbracket M \rrbracket_\Phi \approx \llbracket N \rrbracket_\Phi$. Let

$$\begin{aligned} C^1_M &= \{K_5\}_{K_2} \\ C^2_M &= \left\{(\{001\}_{K_3}, \{K_6\}_{K_5})\right\}_{K_5} \\ C^3_M &= \{K_6\}_{K_5} \\ C^4_M &= \{0\}_{K_6}. \end{aligned}$$

M is a pair of two expressions: $M = (M_1, M_2)$. Then, since $\llbracket (M_1, M_2) \rrbracket_\Phi$ = $\llbracket N \rrbracket_\Phi$, condition (i) of the lemma ensures that N must be a pair too: $N = (N_1, N_2)$. Then, since

$$\llbracket M_1 \rrbracket_\Phi = \pi^1_{\overline{\textbf{strings}} \times \overline{\textbf{strings}}} \circ [\cdot,\cdot]^{-1}(\llbracket M \rrbracket_\Phi),$$

and

$$\llbracket N_1 \rrbracket_\Phi = \pi^1_{\overline{\textbf{strings}} \times \overline{\textbf{strings}}} \circ [\cdot,\cdot]^{-1}(\llbracket N \rrbracket_\Phi)$$

(where $\pi^1_{\overline{\mathbf{strings}}\times\overline{\mathbf{strings}}}$ denotes projection onto the first component of $\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}$), and since $\approx$ is assumed to be preserved by depairing and projecting, it follows that

$$[\![M_1]\!]_\Phi \approx [\![N_1]\!]_\Phi.$$

Therefore, since M_1 is a pair, N_1 must be a pair too. We recursively apply this argument and this way we conclude, that the non-encrypted pairs in M are in the same position as the non-encrypted pairs in N, hence

$$\mathcal{B}(M) = \mathcal{B}(N).$$

It also follows then, that

$$\mathcal{T}_0(M) = \Big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\Big) \times \Big(\big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\big) \times \overline{\mathbf{strings}}\Big) = \mathcal{T}_0(N).$$

At this point, we know that N has the form

$$N = \Big(\big(N_3, N_4\big), \big((N_5, N_6), N_7\big)\Big)$$

Now, we took C^1_M to be $\{K_5\}_{K_2}$, the corresponding string, which is a cipher, is located in the last component of $\mathcal{T}_0(M)$. The key string that decrypts this cipher is located in the third component of $\mathcal{T}_0(M)$. Hence

$$\mathcal{G}_0(C^1_M, M) = \pi^5_{\mathcal{T}_0(M)}$$

and

$$\mathcal{G}_0(C^1_{M,\mathrm{key}}, M) = \pi^3_{\mathcal{T}_0(M)}.$$

But then, since $\pi^i_{\mathcal{T}_0(M)}$ preserves $\approx$, it follows that

$$\pi^3_{\mathcal{T}_0(M)}(\mathcal{B}(M)([\![M]\!]_\Phi)) \approx \pi^3_{\mathcal{T}_0(M)}(\mathcal{B}(N)([\![N]\!]_\Phi)),$$

and

$$\pi^5_{\mathcal{T}_0(M)}(\mathcal{B}(M)([\![M]\!]_\Phi)) \approx \pi^5_{\mathcal{T}_0(M)}(\mathcal{B}(N)([\![N]\!]_\Phi)).$$

It is also true that

$$\pi^3_{\mathcal{T}_0(N)} = \mathcal{G}_0(N_5, N).$$

But

$$\mathcal{G}_0(C^1_{M,\mathrm{key}}, M)(\mathcal{B}(M)([\![M]\!]_\Phi)) = [\![K_2]\!]_\Phi,$$

and

$$\mathcal{G}_0(N_5, N)(\mathcal{B}(N)([\![N]\!]_\Phi)) = [\![N_5]\!]_\Phi,$$

so

$$[\![N_5]\!]_\Phi \approx [\![K_2]\!]_\Phi,$$

and hence, by the assumption (i) of the lemma, it follows that N_5 must also be a key, let us denote it with L_1. Similarly,

$$\pi^5_{\mathcal{T}_0(N)} = \mathcal{G}_0(N_7, N),$$

but then
$$[\![N_7]\!]_\Phi \approx [\![\{K_5\}_{K_2}]\!]_\Phi,$$
and therefore N_7 must be a cipher: $N_7 = \{N'\}_L$ for some expression N' and key L. To get that $L = L_1$, consider

$$(\pi^3_{\mathcal{I}_0(M)}, \pi^5_{\mathcal{I}_0(M)}) \circ \mathcal{B}(M)([\![M]\!]_\Phi) = [\![(K_2, \{K_5\}_{K_2})]\!]_\Phi$$

and
$$(\pi^3_{\mathcal{I}_0(N)}, \pi^5_{\mathcal{I}_0(N)}) \circ \mathcal{B}(N)([\![N]\!]_\Phi) = [\![(L_1, \{N'\}_L)]\!]_\Phi.$$
From this, since the left-hand sides are equivalent, we conclude that

$$[\![(K_2, \{K_5\}_{K_2})]\!]_\Phi \approx [\![(L_1, \{N'\}_L)]\!]_\Phi,$$

which means by condition (ii) of the lemma that

$$L = L_1.$$

Therefore, if we define C^1_N as $\{N'\}_L$, then these terms and the keys that decrypt them are also in the same position, so
$$\mathcal{D}_1(M) = \mathcal{D}_1(N).$$
Remember from example 2.19, that $\mathcal{D}_1(M) = \mathcal{D}_1(N)$ does the following:

$$\mathcal{D}_1(M)\big((x_1, x_2), ((x_3, x_4), x_5)\big) = \begin{cases} \Big((x_1, x_2), \Big((x_3, x_4), \big(x_3, 0, \mathcal{B}(\{K_5\}_{K_2})(\mathcal{D}(x_3, x_5))\big)\Big)\Big) & \\ & \text{if } (x_3, x_5) \in \mathrm{Dom}_{\mathcal{D}} \\ \big((x_1, x_2), ((x_3, x_4), (0, 0, 0)\big) & \text{otherwise,} \end{cases}$$

so if x is sampled from $[\![M]\!]_\Phi$ or $[\![N]\!]_\Phi$, then $\mathcal{D}_1(M)(\mathcal{B}(M)x) = \mathcal{D}_1(N)(\mathcal{B}(N)x)$ has the form

$$\Big((x_1, x_2), \Big((x_3, x_4), \big(x_3, 0, x_6\big)\Big)\Big),$$

and

$$\mathcal{T}_1(M) = \mathcal{T}_1(N) = \Big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\Big) \times \Big(\big(\overline{\mathbf{strings}} \times \overline{\mathbf{strings}}\big) \times \big(\overline{\mathbf{strings}} \times \{0\} \times \overline{\mathbf{strings}}\big)\Big)$$

Then, we continue this process until we show that $\mathcal{D}_4(M) = \mathcal{D}_4(N)$.

2.5.6 Completeness

We finally present our completeness result. Condition (ii) is equivalent to what the authors in [HG03] call weak key-authenticity. Observe, that the issue of key-cycles never rise throughout the proof.

The proof consists of two separate parts. In the first, it is shown that conditions (i) and (ii) imply that if M and N are valid expressions and $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$, then there is a key-renaming σ, such that apart from the boxes, everything else in the patterns of M and $N\sigma$ is the same, and the boxes in the two patterns must be in the same positions. Moreover, condition (iii) implies that picking any two boxes of the pattern of $N\sigma$, there is a key-renaming σ_1 such that applying it to the indexes of these boxes, we obtain the corresponding boxes in the pattern of M. Then the theorem follows, if we prove that using these pairwise equivalences of the boxes, we can construct a σ' that leaves the keys of $N\sigma$ outside the boxes untouched, and it maps the indexes of all the boxes of $N\sigma$ into the indexes of the boxes of M.

Theorem 2.19. *Let $\Delta = (\textbf{\textit{Exp}}_{\mathcal{V}}, \equiv_{\mathbf{K}}, \equiv_{\mathbf{C}})$ be a formal logic for symmetric encryption, such that $\equiv_{\mathbf{C}}$ is proper and that $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent. Let Φ be an interpretation in $\Pi = (\{\mathcal{K}_i\}_{i\in I}, \mathcal{E}, \mathcal{D}, \approx)$. Completeness for Φ holds, if and only if the following conditions are satisfied: For any $K, K', K'' \in$ **Keys**, $B \in$ **Blocks**, $M, M', N \in \textbf{\textit{Exp}}_{\mathcal{V}}$,*

(i) *no pair of $[\![K]\!]_\Phi$, $[\![B]\!]_\Phi$, $[\![(M,N)]\!]_\Phi$, $[\![\{M'\}_{K'}]\!]_\Phi$ is equivalent with respect to $\approx$; that is, keys, blocks, pairs, encryption terms are distinguishable,*

(ii) *if $[\![(K, \{M\}_K)]\!]_\Phi \approx [\![(K'', \{M'\}_{K'})]\!]_\Phi$, then $K' = K''$,*

(iii) *for any two pairs of valid encryption terms $(\{M_1\}_{L_1}, \{M_2\}_{L_2})$ and $(\{N_1\}_{L_1'}, \{N_2\}_{L_2'})$, we have that*

$$[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_{L_1'}, \{N_2\}_{L_2'})]\!]_\Phi$$

implies

$$(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong (\{N_1\}_{L_1'}, \{N_2\}_{L_2'}).$$

Proof. The only if part is trivial. In order to prove the if part, consider two expressions M and N such that $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. By condition (i) and (ii), Lemma 2.18 is applicable, so, $c(M) = c(N)$,

$$\mathcal{D}(M)([\![M]\!]_\Phi) \approx \mathcal{D}(N)([\![N]\!]_\Phi),$$

and

$$\mathcal{T}_{c(M)}(M) = \mathcal{T}_{c(N)}(N).$$

In each entry of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, the distribution corresponds either to the interpretation of a key, or of a block, or of an undecryptable cipher (*i.e.* one that corresponds to a box). Naturally, the same blocks must be in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$, because the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, and because of condition (i). Hence, the patterns of M and N contain the same blocks in the same positions. Moreover, since $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, the entries in $\mathcal{T}_{c(M)}(M)$ and in $\mathcal{T}_{c(N)}(N)$ containing strings sampled from key generation must be in the same places because of (i) again. Furthermore, the indistinguishability of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$ also implies that repetitions of a key generation outcome must occur in the same positions of $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$ as well. (This is a consequence of the properties of key-generation in definition 2.29.) Therefore the key symbols in the patterns of M and N change together, so it is possible to rename the recoverable keys of N (with a $\equiv_{\mathbf{K}}$ preserving function σ so that the keys in the pattern of $N\sigma$ are the same as the keys in the pattern of M.

Since the distributions of $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, condition (i) implies that the undecryptable ciphers occur in exactly the same entries in $\mathcal{T}_{c(M)}(M)$ and $\mathcal{T}_{c(N)}(N)$. This means, that in the pattern of M and N, the boxes appear in the same position. This together with the conclusions of the previous paragraph means, that apart from the boxes, everything else in the pattern of M and of $N\sigma$ must be the same. By replacing N with $N\sigma$, we can assume from now on that the recoverable keys of N and M are identical, and that the pattern of M and N are the same outside the boxes. Therefore, we only have to show that there is a key renaming σ' that carries the boxes of N into the boxes of M without changing the recoverable keys.

Suppose that there are l boxes altogether in the pattern of M (and hence in the pattern of N). Let $\{M_1\}_{L_1}$, $\{M_2\}_{L_2}$,..., $\{M_l\}_{L_l}$ be the l undecryptable terms in M that turn into boxes (in M) and $\{N_1\}_{L'_1}$, $\{N_2\}_{L'_2}$, ..., $\{N_l\}_{L'_l}$ the corresponding undecryptable terms in N. We denote by μ_i and ν_i the equivalence classes of $\{M_i\}_{L_i}$ and $\{N_i\}_{L'_i}$, respectively, with respect to $\equiv_{\mathbf{C}}$. Then, as we showed above, we have that for $i, j \leq l$, $i \neq j$,

$$[\![(\{M_i\}_{L_i}, \{M_j\}_{L_j})]\!]_\Phi \approx [\![(\{N_i\}_{L'_i}, \{N_j\}_{L'_j})]\!]_\Phi$$

holds since $\mathcal{D}(M)([\![M]\!]_\Phi)$ and $\mathcal{D}(N)([\![N]\!]_\Phi)$ are indistinguishable, and thus, by condition (iii),

$$(\{M_i\}_{L_i}, \{M_j\}_{L_j}) \cong (\{N_i\}_{L'_i}, \{N_j\}_{L'_j}).$$

So, by definition of $\cong$, there exists a key-renaming $\sigma_{i,j}$ such that

$$(\Box_{\mu_i}, \Box_{\mu_j}) = (\Box_{\sigma_{i,j}(\nu_i)}, \Box_{\sigma_{i,j}(\nu_j)}),$$

that is, there exists a key-renaming $\sigma_{i,j}$ such that

$$\mu_i = \sigma_{i,j}(\nu_i) \text{ and } \mu_j = \sigma_{i,j}(\nu_j). \tag{2.10}$$

Consider now the class $\mathfrak{C} = \{\{N_i\}_{L'_i}\}_{i=1}^{l}$. Since we assumed by hypothesis that $\equiv_{\mathbf{C}}$ is proper, by Proposition 2.13 (using $S = \textit{R-Keys}(N)$ and noticing that $L'_k \notin \textit{R-Keys}(N)$) we have that for each ν_k, equivalence class of $\{N_k\}_{L'_k}$, there is a representative C_{ν_k} such that:

(i) $\textit{Keys}(C_{\nu_k}) \cap \textit{R-Keys}(N) = \emptyset$,

(ii) $L'_m \not\sqsubseteq C_{\nu_k}$ for all $m \in \{1, 2, \ldots, l\}$,

(iii) if $\nu_{k_1} \neq \nu_{k_2}$, $|(\nu_{k_1})_{\text{key}}| \neq \infty$ and $|(\nu_{k_2})_{\text{key}}| \neq \infty$, then $\textit{Keys}(C_{\nu_{k_1}}) \cap \textit{Keys}(C_{\nu_{k_2}}) \neq \emptyset$ if and only if $(\nu_{k_1})_{\text{key}} = (\nu_{k_2})_{\text{key}} = \{K\}$ for some key K, and in that case

1. $\textit{Keys}(C_{\nu_{k_1}}) \cap \textit{Keys}(C_{\nu_{k_2}}) = \{K\}$,
2. $C_{\nu_{k_1}}$ and $C_{\nu_{k_2}}$ are both of the form $\{\cdot\}_K$, and
3. $K \not\sqsubseteq C_{\nu_{k_1}}$, $K \not\sqsubseteq C_{\nu_{k_2}}$.

(iv) if $\nu_{k_1} \neq \nu_{k_2}$ and either $|(\nu_{k_1})_{\text{key}}| = \infty$ or $|(\nu_{k_2})_{\text{key}}| = \infty$, then $\mathit{Keys}(C_{\nu_{k_1}}) \cap \mathit{Keys}(C_{\nu_{k_2}}) = \emptyset$.

We now define the key-renaming function τ that leaves the recoverable keys of M (and N) untouched but that maps the boxes in the pattern of N to the corresponding boxes in the pattern of M. This definition is done by induction.

Induction Basis: Let us start by defining τ_2.[3] Since we assumed that $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$ are independent, it is possible to modify $\sigma_{1,2}$ such that the resulting renaming function τ_2 that we get leaves

$$S_2 = \left(\bigcup_{i=3}^{l} \mathit{Keys}(C_{\nu_i}) \cup \textit{R-Keys}(N)\right) \setminus (\mathit{Keys}(C_{\nu_1}) \cup \mathit{Keys}(C_{\nu_2}))$$

untouched and is such that

$$\tau_2(\nu_1) = \sigma_{1,2}(\nu_1) \text{ and } \tau_2(\nu_2) = \sigma_{1,2}(\nu_2).$$

If we combine the previous equations with (2.10) we have that

$$\tau_2(\nu_1) = \sigma_{1,2}(\nu_1) = \mu_1$$

and

$$\tau_2(\nu_2) = \sigma_{1,2}(\nu_2) = \mu_2.$$

Induction Hypothesis: Suppose now that we have defined τ_k in a such a way that τ_k leaves the keys in

$$S_k = \left(\bigcup_{i=k+1}^{l} \mathit{Keys}(C_{\nu_i}) \cup \textit{R-Keys}(N)\right) \setminus \left(\bigcup_{i=1}^{k} \mathit{Keys}(C_{\nu_i})\right), \tag{2.11}$$

untouched and is such that

$$\tau_k(\nu_i) = \mu_i \text{ for all } i \leq k.$$

Inductive Step: There are two cases. First suppose that $\nu_{k+1} = \nu_i$ for some $i \leq k$. In this case, we define $\tau_{k+1} = \tau_k$. It is obvious that τ_{k+1} leaves the keys of

$$S_{k+1} = \left(\bigcup_{i=k+2}^{l} \mathit{Keys}(C_{\nu_i}) \cup \textit{R-Keys}(N)\right) \setminus \left(\bigcup_{i=1}^{k+1} \mathit{Keys}(C_{\nu_i})\right),$$

untouched and is such that

$$\tau_{k+1}(\nu_i) = \mu_i \text{ for all } i \leq k+1,$$

since $C_{\nu_{k+1}} = C_{\nu_i}$ and $\nu_{k+1} = \nu_i$.

In the other case, suppose that $\nu_{k+1} \neq \nu_i$ for all $i \leq k$. Consider now the substitution $\sigma_{j,(k+1)}$ with $j \leq k$. By (2.10) we have that

$$\mu_j = \sigma_{j,(k+1)}(\nu_j)$$

[3]We define the base case as $n = 2$ to avoid some cumbersome notation.

and

$$\mu_{k+1} = \sigma_{j,(k+1)}(\nu_{k+1}). \tag{2.12}$$

Since $\equiv_{\mathbf{C}}$ and $\equiv_{\mathbf{K}}$ are independent, considering

$$S = \left(\bigcup_{i=1}^{l} \mathit{Keys}(C_{\nu_i}) \cup \tau_k \left(\bigcup_{i=1}^{l} \mathit{Keys}(C_{\nu_i}) \right) \cup \mathit{R\text{-}Keys}(N) \right) \setminus \mathit{Keys}(C_{\nu_{k+1}})$$

and $\mathfrak{C} = \{C_{\nu_{k+1}}\}$, we have that it is possible to modify $\sigma_{j,(k+1)}$ to σ^* such that

$$\sigma^*(K) = K \text{ for all } K \in S$$

and

$$\sigma^*(\nu_{k+1}) = \sigma_{j,(k+1)}(\nu_{k+1}).$$

Using (2.12), we can rewrite the previous equation as

$$\sigma^*(\nu_{k+1}) = \sigma_{j,(k+1)}(\nu_{k+1}) = \mu_{k+1}.$$

Thus, we have two substitutions, τ_k and σ^* such that

$$\tau_k(\nu_i) = \mu_i \text{ for all } i \leq k \tag{2.13}$$

and

$$\sigma^*(\nu_{k+1}) = \mu_{k+1}. \tag{2.14}$$

Our goal now is to combine these two substitutions into one substitution τ_{k+1} such that

$$\tau_{k+1}(\nu_i) = \mu_i \text{ for all } i \leq k+1, \tag{2.15}$$

and that leaves untouched the keys in

$$S_{k+1} = \left(\bigcup_{i=k+2}^{l} \mathit{Keys}(C_{\nu_i}) \cup \mathit{R\text{-}Keys}(N) \right) \setminus \left(\bigcup_{i=1}^{k+1} \mathit{Keys}(C_{\nu_i}) \right). \tag{2.16}$$

We can immediately notice that by definition, τ_k only changes the keys in $\left(\bigcup_{i=1}^{k} \mathit{Keys}(C_{\nu_i}) \right)$ (recall (2.11)) and that σ^* only alters the keys in $\mathit{Keys}(C_{\nu_{k+1}})$, thus ensuring (2.16). We also notice that from (2.13) and (2.14), (2.15) follows. So, if it is possible to "merge" the two substitutions, the result follows. We do this by showing that the two substitutions are compatible. We show that if both substitutions change the value of one key K, then they change it to the same value, that is, we show that if for a key K, $\tau_k(K) \neq K$ and $\sigma^*(K) \neq K$ then $\tau_k(K) = \sigma^*(K)$.

Suppose that both τ_k and σ^* change the value of a key K'. Then, by the definition of the two substitutions,

$$K' \in \left(\bigcup_{i=1}^{k} \mathit{Keys}(C_{\nu_i}) \right) \cap \mathit{Keys}(C_{\nu_{k+1}}),$$

that is

$$K' \in \textit{Keys}(C_{\nu_i}) \cap \textit{Keys}(C_{\nu_{k+1}}), \tag{2.17}$$

for some $i \in \{1, \ldots, k\}$. By the way we constructed the representatives C_{ν_k} we have that for any two different equivalence classes ν_{k_1} and ν_{k_2},

$$\textit{Keys}(C_{\nu_i}) \cap \textit{Keys}(C_{\nu_{k+1}}) = \emptyset$$

(whenever $|(\nu_i)\mathrm{key}| = \infty$ or $|(\nu_{k+1})\mathrm{key}| = \infty$) or

$$\textit{Keys}(C_{\nu_{k_1}}) \cap \textit{Keys}(C_{\nu_{k_2}}) \neq \emptyset \text{ if and only if } (\nu_{k_1})_{\mathrm{key}} = (\nu_{k_2})_{\mathrm{key}} = \{K\},$$

and in that case

$$\textit{Keys}(C_{\nu_{k_1}}) \cap \textit{Keys}(C_{\nu_{k_2}}) = \{K\}.$$

Applying these results to (2.17) we have that

$$(\nu_i)_{\mathrm{key}} = (\nu_{k+1})_{\mathrm{key}} = \{K'\}. \tag{2.18}$$

Since $\{N_i\}_{L'_i} \in \nu_i$ and $\{N_{k+1}\}_{L'_{k+1}} \in \nu_{k+1}$, we have that $L'_i \in (\nu_i)_{\mathrm{key}}$ and $L'_{k+1} \in (\nu_{k+1})_{\mathrm{key}}$, and using (2.18) it follows that

$$K' = L'_i = L'_{k+1}. \tag{2.19}$$

We just proved that the only key that both τ_k and σ^* change at the same time is K' so we just need to prove that they change it to the same value (in order to be compatible), that is, $\tau_k(K') = \sigma^*(K')$.

By (2.18) we have that $|(\nu_{k+1})_{\mathrm{key}}| = 1$ and so, using Proposition 2.12 and (2.14) it follows that

$$|(\mu_{k+1})_{\mathrm{key}}| = |\sigma^*((\nu_{k+1})_{\mathrm{key}})| = 1.$$

Since $\{M_{k+1}\}_{L_{k+1}} \in \mu_{k+1}$, we have that $L_{k+1} \in (\mu_{k+1})_{\mathrm{key}}$. So, from the previous equation and (2.18) it follows that

$$\sigma^*(K') = \sigma^*(L'_{k+1}) = L_{k+1}. \tag{2.20}$$

If we apply the same reasoning to ν_i and τ_k, again by (2.18) we have that $|(\nu_i)_{\mathrm{key}}| = 1$ and so, using Proposition 2.12 and (2.13) it follows that

$$|(\mu_i)_{\mathrm{key}}| = |\tau_k((\nu_i)_{\mathrm{key}})| = 1.$$

Since $\{M_i\}_{L_i} \in \mu_i$, we have that $L_i \in (\mu_i)_{\mathrm{key}}$. So, from the previous equation and (2.18) it follows that

$$\tau_k(K') = \tau_k(L'_i) = L_i. \tag{2.21}$$

Now consider the substitution $\sigma_{i,(k+1)}$. By (2.10) we have that

$$\mu_i = \sigma_{i,(k+1)}(\nu_i) \text{ and } \mu_{k+1} = \sigma_{i,(k+1)}(\nu_{k+1}).$$

Using (2.18) and Proposition 2.12 it follows that

$$|(\mu_i)_{\mathrm{key}}| = |\sigma_{i,(k+1)}((\nu_i)_{\mathrm{key}})| = 1 \text{ and } |(\mu_{k+1})_{\mathrm{key}}| = |\sigma_{i,(k+1)}((\nu_{k+1})_{\mathrm{key}})| = 1.$$

As said before, $L_i \in (\mu_i)_{\mathrm{key}}$, $L'_i \in (\nu_i)_{\mathrm{key}}$, $L_{k+1} \in (\mu_{k+1})_{\mathrm{key}}$ and $L'_{k+1} \in (\nu_{k+1})_{\mathrm{key}}$ and so

$$\sigma_{i,(k+1)}(L'_i) = L_i \text{ and } \sigma_{i,(k+1)}(L'_{k+1}) = L_{k+1}.$$

Combining this with (2.19), since $L'_i = L'_{k+1}$, we have that

$$L_i = L_{k+1} \tag{2.22}$$

and so by (2.21), (2.22), and (2.20)

$$\tau_k(K') = L_i = L_{k+1} = \sigma^*(K').$$

Thus for any key K' such that both τ_k and σ^* change the value, they are compatible. We then define τ_{k+1} as

$$\tau_{k+1}(K) = \begin{cases} \sigma^*(K) & \text{if } K \in \textit{Keys}(C_{\nu_{k+1}}) \\ \tau_k(K) & \text{otherwise} \end{cases}.$$

Note that by definition of τ_l, it does not change the keys in $S_l = \textit{R-Keys}(N) \setminus \left(\bigcup_{i=1}^{l} \textit{Keys}(C_{\nu_i})\right)$ but, by properness, we have that $\textit{Keys}(C_{\nu_i}) \cap \textit{R-Keys}(N) = \emptyset$ for all $1 \leq i \leq l$ which implies that τ_l does not change the keys in $\textit{R-Keys}(N)$.

The substitution τ that satisfies the required properties, i.e., that leaves the recoverable keys of M and N untouched, but maps the boxes of the pattern of N into the corresponding boxes in the pattern of M, is defined as τ_l (l is the number of boxes in the pattern of M) and that is what we needed to complete the proof. □

Remark 4. Observe, that condition (iii) of the theorem is trivially satisfied when there is only one box, that is, when all encryption terms are equivalent under $\equiv_{\mathbf{C}}$. Also, if completeness holds for a certain choice of $\equiv_{\mathbf{C}}$, then, if $\equiv'_{\mathbf{C}}$ is such that $M \equiv_{\mathbf{C}} N$ implies $M \equiv'_{\mathbf{C}} N$—i.e. when $\equiv'_{\mathbf{C}}$ results fewer boxes—then completeness holds for $\equiv'_{\mathbf{C}}$ as well. Therefore, we can say, that the key to completeness is not to have too many boxes.

Example 2.22 (Completeness for Type-1 and Type-2 Encryption Schemes). The completeness part of our earlier theorems for type-1 and type-2 encryption schemes are clearly special cases of this theorem, because the formal language we introduced for these schemes were such that $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent, and the conditions of the theorems are analogous.

Example 2.23 (Completeness for One-Time Pad). The formal logic for OTP that we presented in Section 2.4 is such that $\equiv_{\mathbf{C}}$ is proper and $\equiv_{\mathbf{K}}$ and $\equiv_{\mathbf{C}}$ are independent. Furthermore, condition (i) of Theorem 2.19 is satisfied due to the tagging we presented in Section 2.4. Condition (ii) is also satisfied because of the tagging: the reason ultimately is that decrypting with the wrong key will sometimes result invalid endings. Condition (iii) is also satisfied, since the pairs of encryption terms must be encrypted with different keys (in OTP, we cannot use the keys twice), and the equivalence $[\![(\{M_1\}_{L_1}, \{M_2\}_{L_2})]\!]_\Phi \approx [\![(\{N_1\}_{L'_1}, \{N_2\}_{L'_2})]\!]_\Phi$ implies that the corresponding lengths in the two encryption terms must be the same: $l(\{M_1\}_{L_1}) = l(\{N_1\}_{L'_1})$ and $l(\{M_2\}_{L_2}) = l(\{N_2\}_{L'_2})$ which implies $(\Box_{l(\{M_1\}_{L_1})}, \Box_{l(\{M_2\}_{L_2})}) = (\Box_{l(\{N_1\}_{L'_1})}, \Box_{l(\{N_2\}_{L'_2})})$. Therefore, $(\{M_1\}_{L_1}, \{M_2\}_{L_2}) \cong (\{N_1\}_{L'_1}, \{N_2\}_{L'_2})$. In conclusion, the formal logic introduced in Section 2.4 is complete.

2.6 Related Work

Work intended to bridge the gap between the cryptographic and the formal models started with several independent approaches, including the work of Lincoln, Mitchell, Mitchell, and Scedrov [LMMS98], Canetti [Can01], Pfitzmann, Schunter and Waidner [PSW00, PW00], and Abadi and Rogaway [AR02]. There are other works such as the one from Guttman, Thayer, and Zuck [GTZ01] aim at the same results but consider specific models or specific properties, specifically consider strand spaces and information-theoretically secure authentication.

A process calculus for analysing security protocols in which protocol adversaries may be arbitrary probabilistic polynomial-time processes is introduced in [LMMS98]. In this framework, which provides a formal treatment of the computational model, security properties are formulated as observational equivalences. Mitchell, Ramanathan, Scedrov, and Teague [MRST06] use this framework to develop a form of process bisimulation that justifies an equational proof system for protocol security.

The approach by Pfitzmann, Schunter and Waidner [PSW00, PW00] starts with a general reactive system model, a general definition of cryptographically secure implementation by simulatability, and a composition theorem for this notion of secure implementation. This work is based on definitions of secure *function* evaluation, *i.e.*, the computation of one set of outputs from one set of inputs [GL91, MR91, Bea91, Can00]. The approach was extended from synchronous to asynchronous systems in [PW01, Can01], which are now known as the *reactive simulatability framework* [PW01, BPW] and the *universal composability framework* [Can01]. A detailed comparison of the two approaches may be found in [DKMR05].

The first soundness result of a formal model under active attacks has been achieved by Backes, Pfitzmann and Waidner [BPW03] within the reactive simulatability framework. Their result comprises arbitrary active attacks and holds in the context of arbitrary surrounding interactive protocols and independently of the goals that one wants to prove about the surrounding protocols; in particular, property preservation theorems for the simulatability have been proved, *e.g.,* for integrity and secrecy [BI03, BP05]. While the original result in [BPW03] considered public-key encryption and digital signatures, the soundness result was extended to symmetric authentication and to symmetric encryption in [BPW05] and [BP04b], respectively.

Another way of trying to bridge the gap between the two models, the one that we follow in this work, was proposed by Abadi and Rogaway [AR02]. In this framework, formal terms with nested operations are considered specifically for symmetric encryption, the adversary is restricted to passive eavesdropping, and the security goals are formulated as indistinguishability of terms. They show that sufficiently strong cryptography enforced computational soundness for a notion of formal equivalence. From this, many other results followed: Abadi and Jürjens [AJ01] extend this result from terms to more general programs. Bana [Ban04] and Adão, Bana, and Scedrov [ABS05] extend the original Abadi-Rogaway result to weaker encryption schemes, while Laud and Corin [LC03] do the same for composite keys. These two extensions are orthogonal: the former extends the applicability of the result to other encryption schemes (e.g., encryption schemes that reveal the length of the underlying plaintext) while the latter extends the set of expressions of the symbolic model.

Herzog, Liskov, and Micali [HLM03] demonstrate soundness for non-malleability properties, and Herzog [Her04] later shows that this soundness for non-malleability is in fact implied by soundness of indistinguishability. The extension of this trace-based approach to active adversaries was done by Micciancio and Warinschi [MW04b] where they show soundness for mutual-authentication properties in the presence of active adversaries. This result is a simpler abstraction than [BPW03] and thus it only addresses a restricted class of protocols.

This trace-based approach, in spite of more restrictive, still allows some extensions such as Micciancio and Panjwani [MP05], soundness of a group-key distribution protocol in the presence of a CPA-secure scheme, Cortier and Warinschi [CW05], use of automated tools to prove that symbolic integrity and secrecy proofs are sound with respect to the computational model in the case of protocols that use nonces, signatures, asymmetric encryption and allow ciphertext forwarding.

Another extension to asymmetric encryption, but still under passive attacks, is in [HLM03]. Asymmetric encryption under active attacks is considered in [Her02] in the random oracle model. Laud [Lau04] has subsequently presented a cryptographic underpinning for a formal model of symmetric encryption under active attacks. His work enjoys a direct connection with a formal proof tool, but it is specific to certain confidentiality properties and restricts the surrounding protocols to straight-line programs in a specific language.

Recently, there has been concurrent and independent work on linking symbolic and cryptographic secrecy properties. Cortier and Warinschi [CW05] have shown that symbolically secret nonces are also computationally secret, *i.e.*, indistinguishable from a fresh random value given the view of a cryptographic adversary. Backes and Pfitzmann [BP05] and Canetti and Herzog [CH06] have established new symbolic criteria that suffice to show that a key is cryptographically secret. Backes and Pfitzmann formulate this as a property preservation theorem from the formal model to a concrete implementation while Canetti and Herzog link their criteria to ideal functionalities for mutual authentication and key exchange protocols.

The first cryptographically sound security proofs of the Needham-Schroeder-Lowe protocol have been presented concurrently and independently in [BP04a] and [War03]. While the first paper conducts the proof within a deterministic, symbolic framework, the proof in the second paper is done from scratch in the cryptographic approach; on the other hand, the second paper proves stronger properties and further shows that chosen-plaintext-secure encryption is insufficient for the security of the protocol.

Impagliazzo and Kapron [IK03] suggest a formal logic for reasoning about probabilistic polynomial-time indistinguishability. Datta, Derek, Mitchell, Shmatikov, and Turuani [DDM$^+$05] describe a cryptographically sound formal logic for proving protocol security properties without explicitly reasoning about probability, complexity, or the actions of a malicious attacker.

Regarding completeness, Micciancio and Warinschi [MW04a] show that a sufficiently strong encryption scheme enforces completeness for indistinguishability properties, and later Horvitz and Gligor [HG03] strengthened this result by giving an exact characterisation of the computational requirements on the encryption scheme under which completeness holds. Later, it was shown by Bana [Ban04] and Adão *et al.* [ABS05] that completeness also holds for a more general class of (weaker) encryption systems.

We stress that none of the aforementioned soundness results hold in the presence of key-cycles. The problem of soundness in the presence of key-cycles was already addressed by

Laud [Lau02]. Laud's solution provides soundness in the presence of key-cycles, but does so by weakening the notion of formal equivalence. It is assumed that key-cycles somehow always 'break' the encryption and the formal adversary is strengthened so as to be always able to 'see' inside the encryptions of a key-cycle. Soundness in the presence of key-cycles naturally holds under this assumption, but we feel that the price paid is too high. Formal equivalence should reflect the ability of the formal adversary to distinguish messages, which should in turn reflect the actual extent to which the computational adversary can distinguish messages. It is often unreasonable from a cryptographer's point of view to *a priori* assume that the computational adversary can break all key-cycles. We therefore propose, in this work, to demonstrate soundness in the presence of key-cycles not by weakening encryption in the formal model, as suggested by Laud, but by strengthening it in the computational one.

2.7 Conclusions and Further Work

We have studied expansions of the Abadi-Rogaway logic of indistinguishability for formal cryptographic expressions, considering and solving two weaknesses of the original result.

The first of these weakness was the problem of soundness in the presence of key-cycles. Computational soundness for expressions *without* key-cycles was proved in Abadi and Rogaway [AR02] under the assumption that a computational encryption scheme satisfies a strong version of semantic security (type-0). We have considered a modification of their logic in the case of encryption schemes both which-key revealing and message-length revealing. In the presence of key-cycles, we have proved that the computational soundness property follows from the key-dependent message (KDM) security proposed by Black *et al.* [BRS02]. As far as we know, this is the first time that in order to achieve soundness, the computational model is strengthened instead of the formal model weakened. We have also shown that the computational soundness property neither implies nor is implied by type-0 security, and thus the original Abadi-Rogaway result could not have been demonstrated for key-cycles using the security notions described in their work.

(We also show in Appendix B that the above results can be extended to the public-key setting. In particular we show that the soundness property holds for arbitrary messages (even with key-cycles) in the presence of a KDM-secure encryption scheme, and that the computational soundness property neither implies nor is it implied by security against chosen ciphertext attack, CCA-2. This is in contrast to many previous results where forms of soundness are implied by CCA-2 security.)

The other weakness of the original Abadi-Rogaway result addressed in this dissertation concerned the possibility of leakage of information by an encryption scheme. As said before, the original result assumed a very strong notion of security (type-0) which is not actually achieved by many encryption schemes. Thus, one might wonder if a similar result might be derived for weaker schemes. We have showed that for symmetric encryption, subtle differences between security definitions can be faithfully reflected in the formal symbolic setting. We have introduced a general probabilistic framework which includes both the computational and the information-theoretic encryption schemes as special cases. We have established soundness and completeness theorems in this general framework, as well as new applications to specific settings: an information-theoretic interpretation of formal expressions in One-Time Pad, and also computational interpretations in type-1 (length-revealing), type-2 (which-key revealing) and type-3 (which-key and length revealing) encryption schemes based on computational complexity.

Our work presents several directions for future research. Independently of any soundness considerations, several questions about KDM security remain unanswered. This is no known implementation of KDM security in the standard model, although there are several natural candidates (*e.g.*, Cramer-Shoup [CS98]). Conversely, there remains to be found a natural (*i.e.*, non-constructed) example of an encryption scheme which is secure in the sense of type-0 (or CCA-2) but is not KDM-secure. Further, even the constructed examples fail to provide KDM security only when presented with key-cycles of length 1. It may in fact be possible that type-0/CCA-2 security implies KDM security when all key-cycles are of length 2 or more.

With regard to soundness in the presence of key-cycles, it seems desirable to extend our results from the passive-adversary setting to that of the active adversary. Also, our results do not completely explore all ‘gaps’ between the two models. We show that the relationship between the formal and computational models requires more than type-0/CCA-2 security. While it demonstrates that KDM security is also necessary, it does not show it to be sufficient—even when conjoined with CCA-2 security (asymmetric encryption). That is, this investigation is not complete; it is more than likely that additional properties will be revealed as soundness is more fully explored.

Also, one might consider various expansions of the formal setting that would allow additional operations such as *xor*, pseudorandom permutations, or exponentiation. Soundness and completeness such richer formal settings would, of course, need exploration. In particular, the definition of patterns appears to be rather subtle in such richer settings. We would also like to understand how our methods fit with the methods of [Mau02].

Lastly, one might consider exploring partial leakage in the setting of asymmetric encryption. One might also extend our methods and investigate formal treatment of other cryptographic primitives. It would be interesting to see if our methods could be combined with the methods of [BPW03, Can01].

Chapter 3

Process Algebras for Studying Security

Process Algebras have been widely used in the study of security of concurrent systems [Mil89, Low95, Low96, AG99, Mil99, AF01, AFG02, BAF05]. In spite of their success in proving security of cryptographic protocols, mainly secrecy and authenticity properties, all these are stated in the so called Dolev-Yao Model, hence no real cryptographic guarantees are achieved.

Another approach is to supplement process calculi with concrete probabilistic or polynomial-time semantics [LMMS98]. Unavoidably, reasoning on processes becomes more difficult.

In this Chapter, we present a process calculus that enjoys both the simplicity of an abstract symbolic model and a concrete (sound and complete) implementation that achieves strong cryptographic guarantees. Our calculus is a variant of the pi calculus with high level security primitives; it provides name mobility, reliable messaging and authentication primitives, but neither explicit cryptography nor probabilistic behaviours.

Taking advantage of concurrency theory, it supports simple reasoning, based on labelled transitions and observational equivalence. We precisely define its concrete implementation in a computational setting. Our implementation relies on standard cryptographic primitives, computational security definitions (CCA2 for encryption [RS91], CMA for signing [GMR88], recalled in Appendix A), and networking assumptions. It also combines typical distributed implementation mechanisms such as abstract machines, marshaling and unmarshaling, multiplexing, and basic communications protocols.

We establish general completeness results in the presence of active probabilistic polynomial-time adversaries, for both trace properties and observational equivalences, essentially showing that high level reasoning accounts for all low-level adversaries.

We illustrate our approach by coding security protocols and establishing their computational correctness by simple formal reasoning.

This Chapter is organised as follows: In Section 3.1, we start by describing the low-level target model as the constraints imposed by this will drive the design of the high-level language. In Section 3.2, we present our high-level language and semantics, and in Section 3.3 we define and illustrate our notion of high-level equivalence. Section 3.4 is devoted to applications. We encode anonymous forwarders in our language and exhibit an example of an electronic payment protocol. We give also as an example the encoding of an initialisation protocol, that is, given any system S that possibly shares names and certificates among principals, we can always find an initial system S° where principals share no information, such that there is a transition from S° to S. Section 3.5 describes our concrete implementation, and in Section 3.6 we state our results. We conclude this Chapter with the discussion of related work, Section 3.7 and pointing out some future directions in which this work may be extended, Section 3.8.

3.1 Low-Level Target Model

Before presenting our language design and implementation, we specify the target systems. We do this, as the design of our language is, in part, driven by the target model. We want to be as abstract as possible, but at the same time we need to faithfully abstract the properties of the computational implementation.

As an example, we want our high-level environments to have the same capabilities as the low-level adversaries, that are probabilistic polynomial-time (PPT) cryptographic algorithms. We follow the conservative assumption that an adversary controls all network traffic: it can intercept, delay, or even block permanently any communication between principals. For that, we cannot guarantee message delivery, nor implement private channels that prevent traffic analysis. Reflecting this in the high-level semantics implies that the simple pi-calculus rule $\overline{c}\langle M\rangle.P \mid c(x).Q \rightarrow P \mid Q\{M/x\}$, which models silent communication is too abstract for our purposes. (Consider P and Q two processes that are implemented in two separate machines connected by a public network, and even if c is a restricted channel, the adversary can simply block all communications.)

We consider systems that consist of a finite number of principals $a, b, c, e, u, v, \ldots \in \mathsf{Prin}$. Each principal a runs its own program, written in our high-level language and executed by the PPT machine M_a defined in Section 3.5. Each machine M_a has two wires, $\mathbf{in}_a$ and $\mathbf{out}_a$, representing a basic network interface. When activated, the machine reads a bitstring from $\mathbf{in}_a$, performs some local computation, then writes a bitstring on $\mathbf{out}_a$ and yields. The machine embeds probabilistic algorithms for encryption, signing, and random-number generation—thus the machine outputs are random variables. The machine is also parameterised by a security parameter $\eta \in \mathbb{N}$—intuitively, the length for all keys—thus these outputs are ensembles of probabilities.

Some of these machines may be corrupted, under the control of the attacker; their implementation is then unspecified and treated as part of the attacker. We let $a, b, c \in \mathcal{H}$ with $\mathcal{H} \subset \mathsf{Prin}$ range over principals that comply with our implementation, and let $\mathsf{M} = (\mathsf{M}_a)_{a\in\mathcal{H}}$ describe our whole system. We denote by e a principal controlled by the adversary ($e \in \mathsf{Prin} \setminus \mathcal{H}$) and by u, v an arbitrary principal in Prin. Of course, when a interacts with $u \in \mathsf{Prin}$, its implementation M_a does not know whether $u \in \mathcal{H}$ or not.

The adversary, A, is a PPT algorithm that controls the network, the global scheduler, and some compromised principals. At each moment, only one machine is active: whenever an adversary delivers a message to a principal, the machine for this principal is activated, runs until completion, and yields an output to the adversary. We have then the following definition:

Definition 3.1 (Run). We define a *run* of A and M with security parameter $\eta \in \mathbb{N}$ as follows:

1. key materials, with security parameter η, are generated for every principal $a \in \mathsf{Prin}$;

2. every M_a is activated with 1^η, the keys for a, and the public keys for all $u \in \mathsf{Prin}$;

3. A is activated with 1^η, the keys for $e \in \mathsf{Prin} \setminus \mathcal{H}$, and the public keys for $a \in \mathcal{H}$;

4. A performs a series of low-level exchanges of the form:

   (a) A writes a bitstring on wire $\mathbf{in}_a$ and activates M_a for some $a \in \mathcal{H}$;

   (b) upon completion of M_a, A reads a bitstring on $\mathbf{out}_a$;

5. A returns a bitstring s, written $s \longleftarrow \mathsf{A}[\mathsf{M}]$.

We keep η implicit whenever possible.

At each Step 4, the adversary A can choose a and compute the bitstring written on $\mathbf{in}_a$ from any previously-received materials, including principal keys and bitstrings collected from previous exchanges.

By design, our low-level runs do not render attacks based on timed properties, such as for instance any observation of the time it takes for each machine to reply. Although the risk of quantitative traffic analysis may be significant, it can be mitigated independently, for instance by sending messages according to a fixed schedule. We leave this discussion outside the scope of this dissertation.

To study the security properties of these runs, we compare systems that consist of machines running on behalf of the same principals $\mathcal{H} \subseteq \mathsf{Prin}$, but with different internal programs and states. Intuitively, two systems are equivalent when no PPT adversary, starting with the information normally given to the principals $e \in \mathsf{Prin} \setminus \mathcal{H}$, can distinguish between their two behaviours, except with negligible probability, Definition A.1. This notion is called *computational indistinguishability* and was introduced by Goldwasser and Micali [GM84]. We state it here in a different but equivalent way.

Definition 3.2 (Low-Level Equivalence). Two systems M^0 and M^1 are indistinguishable, written $\mathsf{M}^0 \approx \mathsf{M}^1$, if for all PPT adversaries A:

$$|\Pr[1 \longleftarrow \mathsf{A}[\mathsf{M}^0]] - \Pr[1 \longleftarrow \mathsf{A}[\mathsf{M}^1]]| \leq \operatorname{neg}(\eta)\,.$$

Our goal is to develop a simpler, higher-level semantics that entails indistinguishability.

3.2 A Distributed Calculus with Principals and Authentication

We now present our high-level language. We successively define terms, patterns, processes, configurations, and systems. We then give their operational semantics. Although some aspects of the design are unusual, the resulting calculus is still reasonably abstract and convenient for distributed programming.

3.2.1 Syntax and Informal Semantics

Definition 3.3 (Names, Terms, Patterns). Let Prin be a finite set of *principal identities*. Let Name be a countable set of *names* disjoint from Prin. Let $\mathtt{f}$ range over a finite number of function symbols, each with a fixed arity $k \geq 0$. We define *terms* and *patterns* by the following grammar:

$V, W ::=$	Terms
x, y	variable
$m, n \in \mathsf{Name}$	name
$a, b, e, u, v \in \mathsf{Prin}$	principal identity
$\mathtt{f}(V_1, \ldots, V_k)$	constructed term (when $\mathtt{f}$ has arity k)
$T, U ::=$	Patterns
$?x$	variable (binds x)
$T\ as\ ?x$	alias (binds x to the term that matches T)
V	constant pattern
$\mathtt{f}(T_1, \ldots, T_k)$	constructed pattern (when $\mathtt{f}$ has arity k)

As usual in process calculi, names and principal identities are atoms, which may be compared with one another but otherwise do not have any structure. Constructed terms represent structured data, much like algebraic data types in ML or discriminated unions in C. They can represent constants and tags (when $k = 0$), tuples, and formatted messages. As usual, we write $\mathtt{tag}$ and (V_1, V_2) instead of $\mathtt{tag}()$ and $\mathtt{pair}(V_1, V_2)$.

Patterns are used for analysing terms and binding selected subterms to variables. For instance, the pattern $(\mathtt{tag}, ?x)$ matches any pair whose first component is $\mathtt{tag}$ and binds x to its second component. We write $\_$ for a variable pattern that binds a fresh variable.

Definition 3.4 (Local Processes). *Local processes* represent the active state of principals, and are defined by the following grammar:

$P, Q, R ::=$	Local processes
V	asynchronous output
$(T).Q$	input (binds $bv(T)$ in Q)
$*(T).Q$	replicated input (binds $bv(T)$ in Q)
$\mathtt{match}\ V\ \mathtt{with}\ T\ \mathtt{in}\ Q\ \mathtt{else}\ Q'$	matching (binds $bv(T)$ in Q)
$\nu n.P$	name restriction ("new", binds n in P)
$P \mid P'$	parallel composition
$\mathbf{0}$	inert process

The asynchronous output V is just a pending message; its data structure is explained below. The input $(T).Q$ waits for an output that matches the pattern T then runs process Q with the bound variables of T substituted by the matching subterms of the output message. The replicated input $*(T).Q$ behaves similarly but it can consume any number of outputs that match T and fork a copy of Q for each of them. The match process runs Q if V matches T, and runs Q' otherwise. The name restriction creates a fresh name n then runs P. Parallel composition represents processes that run in parallel, with the inert process $\mathbf{0}$ as unit.

Free and bound names and variables for terms, patterns, and processes are defined as usual: x is bound in T if $?x$ occurs in T; n is bound in $\nu n.P$; x is free in T if it occurs in T and is not bound in T. An expression is closed when it has no free variables; it may have free names.

Definition 3.5 (Local Contexts). A *local context* is a process with a hole instead of a subprocess. We say that a context is an *evaluation context* when the hole replaces a subprocess P or P' in the grammar of Definition 3.4. If it replaces a subprocess Q or Q' we call it a *guarded context*.

Our language features two forms of authentication, represented as two constructors `auth` and `cert` of arity 3 plus well-formed conditions on their usage in processes.

Definition 3.6 (Authenticated Messages, Certificates). *Authenticated messages* between principals are represented as terms of the form $\mathtt{auth}(V_1, V_2, V_3)$, written $V_1{:}V_2\langle V_3\rangle$, where V_1 is the sender, V_2 the receiver, and V_3 the content. We let M and N range over messages. The message M is *from* a (respectively *to* a) if a is the sender (respectively the receiver) of M.

Certificates issued by principals are represented as terms of the form $\mathtt{cert}(V_1, V_2, V_3)$, written $V_1\{V_2\}_{V_3}$, where V_1 is the issuer, V_2 the content, and V_3 the label.

Labels in certificates reflect cryptographic signature values in their implementation. They are often unimportant (and omitted), since our processes use a constant label `0` in their certificates and ignore labels (using _) in their certificate patterns. Nonetheless, they are necessary because the standard definition of security for signatures (CMA-security, Definition A.6) does not exclude the possibility that the attacker produce different signature values for certificates with identical issuer and content. If we do not include labels in our definition of high-level certificates, we could be excluding attacks.

Example 3.1. Consider a protocol where adversarial principal e receives a certificate $cert_1$ from a, forges a second certificate $cert_2$ using some malleability property of the signing scheme, and then forwards $cert_1$ to b and $cert_2$ to c.

If later e receives $cert_i$ from d, he may discover part of the topology of the network, as $i = 1$ if d is connected to b and $i = 2$ if d is connected to c. If the attack to the protocol depends upon the knowledge of the network, we have an attack.

If we do not account for this possibility in our high-level semantics, that is, use different labels for different certificates, we could never capture this attack as the received certificate by e would be equivalent regardless of $i = 0$ or $i = 1$.

Although both authenticated messages and certificates provide some form of authentication, authenticated messages are delivered at most once, to their designated receiver, whereas certificates can be freely copied and forwarded within messages. Hence, certificates conveniently represent transferable credentials and capabilities. They may be used, for instance, to code decentralised access-control mechanisms.

Example 3.2. As an example, $a{:}b\langle\mathtt{Hello}\rangle$ is an authentic message from a to b with content `Hello`, a constructor with arity 0, for which b (and only b) can verify that it is coming from a

$a\{b, \mathtt{Hello}\}$ is a certificate signed by a with the same subterms that can be sent, received, and verified by any principal.

We let $\phi(V)$ be the set of certificates included in V and let $\phi(V)_X \subseteq \phi(V)$ be those certificates issued by $u \in X$. For instance, we have

$$\phi(a\{0, b\{1\}, c\{2\}\})_{\{a,b\}} = \{b\{1\},\ a\{0, b\{1\}, c\{2\}\}\}$$

Definition 3.7 (Well Formed Process). Let P be a local process. We say that P is *well-formed for* $a \in \mathsf{Prin}$ when:

1. any certificate in P that includes a variable or a bound name is of the form $a\{V_2\}_{\mathsf{0}}$;
2. no pattern in P binds any certificate label; and
3. no pattern used for input in P matches any authenticated message from a.

Condition 1 states that the process may produce new certificates only with issuer a; in addition, the process may contain previously-received certificates issued by other principals. (We do not restrict certificate patterns—a pattern that tests a certificate not available to a will never be matched.) Condition 2 restricts access to labels, so that labels only affect comparisons between certificates. Condition 3 prevents that authenticated messages sent by P be read back by some local input.

Finally, we are now able to define configurations and systems. A *configuration* is an assembly of running principals, each with its own local state, plus an abstract record of the messages intercepted by the environment and not forwarded yet to their intended recipients. A *system* is a top-level configuration plus an abstract record of the environment's knowledge, as a set of certificates previously issued and sent to the environment by the principals in C.

Definition 3.8 (Configurations, Systems). *Configurations* and *systems* are defined by the following grammar:

$C ::=$	configurations
$\quad a[P_a]$	principal a with local state P_a
$\quad M/i$	intercepted message M with index i
$\quad C \mid C'$	distributed parallel composition
$\quad \nu n.C$	name restriction ("new", binds n in C)
$S ::=$	systems
$\quad \Phi \vdash C$	configuration C exporting certificates Φ

and satisfy the following well-formed conditions:

- In configurations, intercepted messages have distinct indices i and closed content M; principals have distinct identities a and well-formed local processes P_a.
- In systems, let $\mathcal{H}$ be the set of identities for all defined principals, called *compliant principals*; intercepted messages are from a to b for some $a, b \in \mathcal{H}$ with $a \neq b$; Φ is a set of closed certificates with label $\mathsf{0}$ such that $\phi(\Phi)_{\mathcal{H}} = \Phi$.

3.2.2 Operational Semantics

We define our high-level semantics in two stages: local reductions between processes, then global labelled transitions between systems and their (adverse) environment. Processes, configurations, and systems are considered up to renaming of bound names and variables.

Local Reductions

We start by defining *structural equivalence*. It represents structural rearrangements for local processes. Intuitively, these rearrangements are not observable (although this is quite hard to implement).

Definition 3.9 (Structural Equivalence for Processes). *Structural equivalence*, written $P \equiv P'$, is defined as the smallest congruence such that:

$$\begin{array}{l}
P \equiv P \,|\, 0 \\
P \,|\, Q \equiv Q \,|\, P \\
P \,|(Q \,|\, R) \equiv (P \,|\, Q) \,|\, R \\
(\nu n.P) \,|\, Q \equiv \nu n.(P \,|\, Q) \text{ when } n \notin \mathit{fn}(Q) \\
\nu m.\nu n.P \equiv \nu n.\nu m.P \\
\nu n.\mathbf{0} \equiv \mathbf{0}
\end{array}$$

Definition 3.10 (Local Reductions, Stable Processes). *Local reduction step*, written $P \rightarrow P'$, represents internal computation between local processes, and is defined as the smallest relation such that

$$\begin{array}{lr}
\text{(LC\small{OMM})} & (T).Q \,|\, T\sigma \rightarrow Q\sigma \\
\text{(LR\small{EPL})} & *(T).Q \,|\, T\sigma \rightarrow Q\sigma \,|\, *(T).Q \\
\text{(LM\small{ATCH})} & \texttt{match}\ T\sigma\ \texttt{with}\ T\ \texttt{in}\ P\ \texttt{else}\ Q \rightarrow P\sigma \\
\text{(LN\small{O}M\small{ATCH})} & \texttt{match}\ V\ \texttt{with}\ T\ \texttt{in}\ P\ \texttt{else}\ Q \rightarrow Q \ \text{ when } V \neq T\sigma \text{ for any } \sigma
\end{array}$$

$$\text{(LP\small{AR}C\small{TX})}\ \frac{P \rightarrow Q}{P \,|\, R \rightarrow Q \,|\, R} \qquad \text{(LN\small{EW}C\small{TX})}\ \frac{P \rightarrow Q}{\nu n.P \rightarrow \nu n.Q} \qquad \text{(LS\small{TRUCT})}\ \frac{P \equiv P' \quad P' \rightarrow Q' \quad Q' \equiv Q}{P \rightarrow Q}$$

where σ ranges over substitutions of closed terms for the variables bound in T.

The local process P is *stable* when it has no local reduction step, written $P \not\rightarrow$. We write $P \twoheadrightarrow Q$ when $P \rightarrow^* \equiv Q$ and $Q \not\rightarrow$.

System Transitions

We define a labelled transition semantics for configurations, then for systems. Each labelled transition, written $S \xrightarrow{\gamma} S'$, represents a single interaction with the adversary. We let α and β range over input and output labels (respectively from and to the adversary), let γ range over labels, and let φ range over series of labels. We write $S \xrightarrow{\varphi} S'$ for a series of transitions with labels φ. We also rely on structural equivalence for configurations,

Definition 3.11 (Structural Equivalence for Configurations). *Structural equivalence for configurations*, written $C \equiv C'$, is defined as the smallest congruence such that:

$$\begin{array}{lll}
C \equiv C' \mid 0 & C \mid C' \equiv C' \mid C & C \mid (C' \mid C'') \equiv (C \mid C') \mid C'' \\
\nu m.\nu n.C \equiv \nu n.\nu m.C & (\nu n.C) \mid C' \equiv \nu n.(C \mid C') \text{ when } n \notin fn(C') & \\
\nu n.a[P] \equiv a[\nu n.P] & &
\end{array}$$

Definition 3.12 (Labels). *Labels* are defined by the following grammar:

$\alpha ::=$	input labels
$\quad (M)$	input of message M
$\quad (i)$	forwarding of intercepted message i
$\beta ::=$	output labels
$\quad \nu n_1 \ldots n_k.M$	output of message M ($n_1, \ldots, n_k \in fn(M)$)
$\quad \nu i.a{:}b$	interception of message i from a to b ($a, b \in \mathcal{H}$)
$\gamma ::=$	single label
$\quad \alpha + \beta$	input or output label
$\varphi ::=$	series of transition labels
$\quad \gamma^*$	

We let $input(\varphi)$ be the series of input labels in φ.

Definition 3.13 (Labelled Transitions for Configurations). *Labelled transitions for configurations* are defined by the following rules:

$$(\textsc{CfgOut})\frac{u \neq a}{a[a{:}u\langle V\rangle \mid Q] \xrightarrow{a:u\langle V\rangle} a[Q]} \qquad (\textsc{CfgIn})\frac{u{:}a\langle V\rangle \mid P \twoheadrightarrow Q \quad u \neq a}{a[P] \xrightarrow{(u:a\langle V\rangle)} a[Q]}$$

$$(\textsc{CfgBlock})\frac{C \xrightarrow{b:a\langle V\rangle} C' \quad i \text{ not in } C}{C \mid a[P] \xrightarrow{\nu i.b:a} C' \mid a[P] \mid b{:}a\langle V\rangle/i} \qquad (\textsc{CfgFwd})\frac{C \xrightarrow{(M)} C'}{C \mid M/i \xrightarrow{(i)} C'}$$

$$(\textsc{CfgPrinCtx})\frac{C \xrightarrow{\gamma} C' \quad \gamma \text{ not from/to } a}{C \mid a[P] \xrightarrow{\gamma} C' \mid a[P]} \qquad (\textsc{CfgMsgCtx})\frac{C \xrightarrow{\gamma} C' \quad i \text{ not in } \gamma}{C \mid M/i \xrightarrow{\gamma} C' \mid M/i}$$

$$(\textsc{CfgOpen})\frac{C \xrightarrow{\beta} C' \quad n \text{ free in } \beta}{\nu n.C \xrightarrow{\nu n.\beta} C'} \qquad (\textsc{CfgNewCtx})\frac{C \xrightarrow{\gamma} C' \quad n \text{ not in } \gamma}{\nu n.C \xrightarrow{\gamma} \nu n.C'}$$

$$(\textsc{CfgStr})\frac{C \equiv D \quad D \xrightarrow{\gamma} D' \quad D' \equiv C'}{C \xrightarrow{\gamma} C'}$$

Rules (CFGOUT) and (CFGIN) represent “intended” interactions with the environment, as usual in an asynchronous pi calculus. They enable local processes for any $a \in \mathcal{H}$ to send messages to other principals u, and to receive their messages. The transition label conveys the complete message content.

Rules (CFGBLOCK) and (CFGFWD) reflect the actions of an active attacker that intercepts messages exchanged between compliant principals, and selectively forwards those messages. In contrast with the (COMM) rule of the pi calculus, they ensure that the environment mediates all communications between principals. The label produced by (CFGBLOCK) signals the message interception; the label conveys partial information on the message content that can intuitively be observed from its wire format: the environment learns that an opaque message is sent by b with intended recipient a. In addition, the whole intercepted message is recorded within the configuration, using a fresh index i. Later on, when the environment performs an input with label (i), Rule (CFGFWD) restores the original message and consumes M/i; this ensures that any intercepted message is delivered at most once.

The local-reduction hypothesis in Rule (CFGIN) demands that all local reductions triggered by the received message be immediately performed, leading to some updated stable process Q. Intuitively, this enforces a transactional semantics for local steps, and prevents any observation of their transient internal state. (Otherwise, the environment may for instance observe the order of appearance of outgoing messages.) On the other hand, any outgoing messages are kept within Q; the environment can obtain all of them via rules (CFGOUT) and (CFGBLOCK) at any time, since those outputs commute with any subsequent transitions.

The rest of the rules for configurations are standard closure rules with regards to evaluation contexts and structural rearrangements: Rule (CFGOPEN) is the scope extrusion rule of the pi calculus that opens the scope of a restricted name included in a message sent to the environment; this rule does not apply to intercepted messages. Rule (CFGPRINCTX) deals with principal a defined in the configuration; condition γ not from a excludes inputs from the environment that would forge a message from a, whereas condition γ not to a excludes outputs that may be transformed by Rule (CFGBLOCK).

Finally, we have a pair of top level rules that deal with the attacker knowledge:

Definition 3.14 (Labelled Transitions for Systems). *Labelled transitions for systems* are defined by the following rules:

$$(\textsc{SysOut})\frac{C \xrightarrow{\beta} C'}{\Phi \vdash C \xrightarrow{\beta} \Phi \cup \phi(\beta)_{\mathcal{H}} \vdash C'} \qquad (\textsc{SysIn})\frac{C \xrightarrow{\alpha} C' \quad \phi(\alpha)_{\mathcal{H}} \subseteq \mathcal{M}(\Phi)}{\Phi \vdash C \xrightarrow{\alpha} \Phi \vdash C'}$$

where $\mathcal{H}$ is the set of principals defined in C and $\mathcal{M}(\Phi) = \{a\{V\}_\ell : a\{V\}_0 \in \Phi\}$ is the set of certificates the attacker might produce from Φ (see Appendix A for the motivation for this rule).

Rule (SYSOUT) filters every output β and adds to Φ the certificates included in β. Rule (SYSIN) filters every input α, and checks that the certificates included in α can be produce from the certificates in Φ.

Our main results are expressed using *normal transitions* between systems.

Definition 3.15 (Stable Systems, Normal Transitions). We say that the system S is *stable* when all local processes of S are stable and S has no output transition. (Informally, S is waiting for any input from the environment.)

We say that a series of transitions $S \xrightarrow{\varphi} S'$ is *normal* when every input transition is followed by a maximal series of output transitions leading to a stable system, that is, $\varphi = \varphi_1\varphi_2 \ldots \varphi_n$ where $\varphi_i = \alpha_i\widetilde{\beta_i}$ for $i = 1..n$, and $S = S_0 \xrightarrow{\varphi_1} S_1 \xrightarrow{\varphi_2} S_2 \ldots \xrightarrow{\varphi_n} S_n = S'$ for some stable systems $S_0, \ldots, S_n$.

Intuitively normality states that each principals outputs all his messages and stays idle until he receives a new input.

Compositionality

By design, our semantics is compositional, as its rules are inductively defined on the structure of configurations. For instance, we obtain that interactions with a principal that is implicitly controlled by the environment are at least as expressive as those with any principal explicited within the system.

If we have $C \mid a[P] \xrightarrow{\alpha} C' \mid a[P']$, then we also have $C^\circ \xrightarrow{\beta} C'^\circ$, where C° and C'° are obtained from C and C', respectively, by erasing the state associated with a: any intercepted messages M/i from a or to a; and any certificate in Φ issued by a. This compositional property yields useful congruence properties for observational equivalence on configurations.

3.2.3 An Abstract Machine for Local Reductions

In preparation to the description of a concrete machine M_a that executes a's local process P_a, we derive a simple algorithm for local reductions. In contrast with our non-deterministic reduction semantics, the algorithm relies on partial normal forms instead of structural equivalence, and it carefully controls the creation of fresh names (to be implemented as random-number generation); it also relies on an explicit scheduler and is otherwise deterministic.

A process P is in *normal form for* a when it is a closed well-formed process such that every subprocess of the form `match` V `with` T `in` Q `else` Q' or $\nu n.Q$ appears only under an input or a replicated input—intuitively, all name creations and matchings are guarded. Let P be in normal form for a. Up to the structural laws for parallel composition, $P \equiv M \mid L \mid G$ where M is a parallel composition of messages sent to other principals, L is a parallel composition of other (local) messages, and G is a parallel composition of inputs and replicated inputs. Concretely, we may represent P as a triple of multisets for M, L, and G.

A *scheduler* is a deterministic algorithm on (L, G) that selects an instance of Rule (LCOMM) or (LREPL) for an output of L and an input (or replicated input) of G, if any, and otherwise reports completion. The reduction algorithm repeatedly calls the scheduler, performs the selected reduction step, then normalises the resulting process by applying Rules (LMATCH) and (LNOMATCH) and lifting name restrictions to the top level (possibly after a renaming). This yields a local process of the form $\nu\widetilde{n}.(M' \mid L' \mid G')$ where $\widetilde{n}$ collect all new name restrictions in evaluation context. By induction on the length of the derivation, one easily check that $P \twoheadrightarrow Q$ if and only if, for some scheduler, the algorithm returns $\widetilde{n}$ and P' in normal form such that $Q \equiv \nu\widetilde{n}.P'$.

A configuration is in normal form when all restrictions are grouped at top-level and every local process is in normal form. Our local reduction strategy can be extended to configurations in normal forms as follows: we perform local reductions as detailed above, then lift any resulting restrictions to the top level of the configuration up to structural equivalence (using $a[\nu\widetilde{n}.P'] \equiv \nu\widetilde{n}.a[P']$).

3.3 High-Level Equivalences and Safety

Now that we have defined labelled transitions that capture our attacker model and implementation constraints, we can apply standard definitions and proof techniques from concurrency theory to reason about systems. Our computational soundness results are useful (and non-trivial) inasmuch as transitions are simpler and more abstract than low-level adversaries. In addition to trace properties (used, for instance, to express authentication properties as correspondences between transitions), we consider equivalences between systems.

Intuitively, two systems are equivalent when their environment observes the same transitions. Looking at immediate observations, we say that two systems S_1 and S_2 *have the same labels* when, if $S_1 \xrightarrow{\gamma} S_1'$ for some S_1' (and the name exported by γ are not free in S_2), then $S_2 \xrightarrow{\gamma} S_2'$ for some S_2', and vice versa. More generally, bisimilarity demands that this remains the case after matching transitions:

Definition 3.16 (Bisimilarity). The relation $\mathcal{R}$ on systems is a labelled simulation when, for all $S_1 \mathcal{R} S_2$, if $S_1 \xrightarrow{\gamma} S_1'$ (and the names exported by γ are not free in S_2) then $S_2 \xrightarrow{\gamma} S_2'$ and $S_1' \mathcal{R} S_2'$. Labelled bisimilarity, written $\simeq$, is the largest symmetric labelled simulation.

In particular, if $\Phi \vdash C \simeq \Phi' \vdash C'$ then C and C' define the same principals, intercepted-message indices, and exported certificates ($\mathcal{M}(\Phi) = \mathcal{M}(\Phi')$).

We also easily verify some congruence properties: our equivalence is preserved by addition of principals, deletion of intercepted messages, and deletion of certificates.

Lemma 3.1.

1. *If $\Phi \vdash C_1 \simeq \Phi \vdash C_2$, then $\Phi \cup \Phi_a \vdash C_1 \,|\, a[P] \simeq \Phi \cup \Phi_a \vdash C_2 \,|\, a[P]$ for any certificates Φ_a issued by a such that the systems are well-formed and $\phi_{\mathcal{H}}(P) \subseteq \Phi$.*
2. *If $\Phi \vdash \nu\widetilde{n}_1.(C_1 \,|\, M_1/i) \simeq \Phi \vdash \nu\widetilde{n}_2.(C_2 \,|\, M_2/i)$, then $\Phi \vdash \nu\widetilde{n}_1.C_1 \simeq \Phi \vdash \nu\widetilde{n}_2.C_2$.*
3. *If $\Phi \cup \{V\} \vdash C_1 \simeq \Phi \cup \{V\} \vdash C_2$ and $V \notin \phi(\Phi)$, then $\Phi \vdash C_1 \simeq \Phi \vdash C_2$.*

Proof. The proof is by bisimulation. We detail the proof of Property 1 of the lemma—the proofs for the other two properties are similar but simpler. For fixed $\mathcal{H} \subset \mathsf{Prin}$ and $a \in \mathsf{Prin} \setminus \mathcal{H}$, we let $\mathcal{R}$ be the relation defined by: if $\Phi \vdash C_1 \simeq \Phi \vdash C_2$, then

$$\Phi_\star \vdash \nu\widetilde{n}.(C_1 \,|\, a[P] \,|\, C_a) \;\mathcal{R}\; \Phi_\star \vdash \nu\widetilde{n}.(C_2 \,|\, a[P] \,|\, C_a)$$

for any names $\widetilde{n}$, configurations C_1, C_2 that define the principals $b \in \mathcal{H}$, local process P, parallel composition C_a of intercepted messages from a or to a, and sets of certificates Φ and $\Phi_\star$ such that the systems are well-formed and

$$\phi_{\mathcal{H}}(\Phi_\star) \cup \phi_{\mathcal{H}}(P) \cup \phi_{\mathcal{H}}(C_a) \quad \subseteq \quad \Phi \tag{3.1}$$

We show that $\mathcal{R}$ is a labelled simulation by case analysis on the transitions of any systems related by $\mathcal{R}$, of the form

$$S_1 = \Phi_\star \vdash \nu\widetilde{n}.(C_1 \,|\, a[P] \,|\, C_a) \xrightarrow{\gamma} S_1' = \Phi_\star' \vdash \nu\widetilde{n}'.(C_1' \,|\, a[P'] \,|\, C_a')$$

Assuming that $S_1 \mathcal{R} S_2$, we establish the existence of a matching transition

$$S_2 = \Phi_\star \vdash \nu\widetilde{n}.(C_2 \,|\, a[P] \,|\, C_a) \xrightarrow{\gamma} S_2' = \Phi_\star' \vdash \nu\widetilde{n}'.(C_2' \,|\, a[P'] \,|\, C_a')$$

such that $S_1' \mathcal{R} S_2'$. We deal with outputs (Rule (SYSOUT)), then inputs (Rule (SYSIN)).

- $\gamma = \nu i.a{:}b$. The transition uses Rule (CFGBLOCK) with index i fresh in S_1 and $b \in \mathcal{H}$ to intercept an output produced by Rule (CFGIN): $a[P] \xrightarrow{\nu\widetilde{m}.a:b\langle V\rangle} a[P']$. Up to renaming, we assume that the names $\widetilde{m}$ are fresh. The index i is also fresh in S_2.

  To obtain a matching transition with $S_1' \mathcal{R} S_2'$, we use this P', we let $C_a' = C_a \,|\, a{:}b\langle V\rangle/i$, $\widetilde{n}' = \widetilde{n}, \widetilde{m}$, and we leave the other parameters unchanged: $C_1' = C_1$, $C_2' = C_2$, $\Phi' = \Phi$, and $\Phi_\star' = \Phi_\star$. Property (3.1) is preserved because $\phi(P') \subseteq \phi(P)$.

- $\gamma = \nu\widetilde{m}.a{:}e\langle V\rangle$ for some $e \notin \mathcal{H} \cup \{a\}$. The transition also uses Rule (CFGIN): we have $a[P] \xrightarrow{\nu\widetilde{m}'.a:b\langle V\rangle} a[P']$ for some fresh names $\widetilde{m}'$. Let $\widetilde{m}'' = fn(V) \cap \widetilde{n}$. We have $\widetilde{n} = \widetilde{n}' \uplus \widetilde{m}''$ and $\widetilde{m} = \widetilde{m}' \uplus \widetilde{m}''$.

  To obtain a matching transition with $S_1' \mathcal{R} S_2'$, we use P' and $\widetilde{n}'$, we let $\Phi_\star' = \Phi_\star \cup \phi_{\mathcal{H}\cup\{a\}}(V)$ and we leave Φ', C_1, and C_2 unchanged. Property (3.1) is preserved, as $\phi_{\mathcal{H}\cup\{a\}}(V) = \phi_{\mathcal{H}}(V) \cup \phi_{\{a\}}(V)$ and $\phi_{\mathcal{H}}(V) \subseteq \phi_{\mathcal{H}}(P) \subseteq \Phi$.

- $\gamma = \nu i.b : a$ for some $b \in \mathcal{H}$. The transition uses Rule (CFGBLOCK) with index i fresh in S_1 and $b \in \mathcal{H}$ to intercept an output produced by Rule (CFGOUT): $C_1 \xrightarrow{\nu\widetilde{m}b:a\langle V\rangle.} C_1'$ for some fresh names $\widetilde{m}$.

  By Rule (SYSOUT), we have $\Phi \vdash C_1 \xrightarrow{\nu\widetilde{m}b:a\langle V\rangle.} \Phi' \vdash C_1'$ where $\Phi' = \Phi \cup \phi_{\mathcal{H}}(V)$.

  By bisimilarity hypothesis $\Phi \vdash C_1 \simeq \Phi \vdash C_2$, we obtain C_2' such that $\Phi \vdash C_2 \xrightarrow{\nu\widetilde{m}b:a\langle V\rangle.} \Phi' \vdash C_2'$ and $\Phi' \vdash C_1' \simeq \Phi' \vdash C_2'$.

  To obtain a matching transition with $S_1' \mathcal{R} S_2'$, we use C_1', C_2', Φ', we let $C_a' = C_a \,|\, b{:}a\langle V\rangle$ and $\widetilde{n}' = \widetilde{n}, \widetilde{m}$, and we leave $\Phi_\star$ and P unchanged.

- $\gamma = \nu\widetilde{m}.b{:}e\langle V\rangle$.

  Similarly, the transition uses (CFGOUT): $\Phi \vdash C_1 \xrightarrow{\nu\widetilde{m}'b:a\langle V\rangle.} \Phi' \vdash C_1'$ where $\Phi' = \Phi \cup \phi_{\mathcal{H}}(V)$ and, by bisimulation hypothesis, we obtain C_2' such that $\Phi \vdash C_2 \xrightarrow{\nu\widetilde{m}b:a\langle M\rangle.} \Phi' \vdash C_2'$ with $\Phi' \vdash C_1' \simeq \Phi' \vdash C_2'$.

  To obtain a matching transition and $S_1' \mathcal{R} S_2'$, we use C_1', C_2', Φ', $\widetilde{n}' = \widetilde{n}, \widetilde{m}$ and we leave C_a, $\Phi_\star$, and P unchanged. Property (3.1) is preserved, since $\phi_{\mathcal{H}}(\phi_{\{a\}}(M)) \subseteq \Phi$ by hypothesis.

- $\gamma = (e{:}u\langle V\rangle)$. We have $\phi_{\mathcal{H}\cup\{a\}}(V) \subseteq \Phi_\star$ by Rule (SYSIN) and $\phi_{\mathcal{H}}(\Phi_\star) \subseteq \Phi$ by Property (3.1), so $\phi_{\mathcal{H}}(V) \subseteq \Phi$. Up to a renaming of $\widetilde{n}$, we assume that the names of V do not clash with $\widetilde{n}$. We distinguish two subcases:

  - if $u = a$, then $a[P] \xrightarrow{\gamma} a[P']$ by Rule (CFGIN).

    We use P' and leave the other parameters unchanged. Property (3.1) is preserved: $\phi_{\mathcal{H}}(P) \subseteq \Phi$ also by Property (3.1), so $\phi_{\mathcal{H}}(P') \subseteq \Phi$ by definition of local reductions and P well-formed for a.

  - otherwise, $u = b$ for some $b \in \mathcal{H}$, and $\Phi \vdash C_1 \xrightarrow{\gamma} \Phi \vdash C_1'$ by Rule (CFGIN).

    By bisimulation hypothesis, we obtain C_2' such that $\Phi \vdash C_2 \xrightarrow{\gamma} \Phi \vdash C_2'$ and $\Phi \vdash C_1' \simeq \Phi \vdash C_2'$.

    We use C_1' and C_2', and leave the other parameters unchanged.

- $\gamma = (i)$. We similarly conclude in each of the following subcases: M/i to a; C_a defines M/i from a; or C_1 defines M/i.

Finally, $\mathcal{R}$ is symmetric by construction, hence $\mathcal{R} \subseteq \simeq$, and $\mathcal{R}$ contains the systems related by the lemma for $\widetilde{n} = \emptyset$, $\Phi_\star = \Phi$, and $C_a = 0$. □

Bounding processes

As we quantify over all local processes, we must at least bound their computational power. Indeed, our language is expressive enough to code Turing machines and, for instance, one can easily write a local process that receives a high-level encoding of the security parameter η (e.g. as a series of η messages) then delays a message output by 2^η reduction steps, or even implements an 'oracle' that performs some brute-force attacks using high level implementations of cryptographic algorithms.

Similarly, we must restrict non-deterministic behaviours. Process calculi often feature non-determinism as a convenience when writing specifications, to express uncertainty as regards the environment. Sources of non determinism include local scheduling, hidden in the associative-commutative laws for parallel composition, and internal choices. Accordingly, abstract properties and equivalences typically only consider the existence of transitions—not their probability. Observable non-determinism is problematic in a computational cryptographic setting, as for instance a non-deterministic process may be used as an oracle to guess every bit of a key in linear time.

In order to bound the complexity of processes (mainly the complexity of reductions) we define a function $\lceil \cdot \rceil$ that computes the high-level structural size of systems, labels and transitions. This is done by structural induction, with for instance $\lceil S \xrightarrow{\beta} S' \rceil = \lceil S \rceil + \lceil \beta \rceil + \lceil S' \rceil + 1$. As for input labels we have that the complexity of $S \xrightarrow{(\alpha)} S'$ accounts also for the internal reductions performed during the transition, that is, $\lceil S \xrightarrow{(\alpha)} S' \rceil = \lceil S \rceil + \lceil \alpha \rceil + \lceil S' \rceil + \lceil u{:}a\langle V \rangle \mid P \twoheadrightarrow Q \rceil + 1$, where $a[P]$ is defined in S and $a[Q]$ is defined in S'. We omit the rest of the details as they are straightforward.

Definition 3.17 (Safe Systems). A system S is *polynomial* when there exists a polynomial p_S and a constant c such that, for any φ, if $S \xrightarrow{\varphi} S'$ then $\lceil S \xrightarrow{\varphi} S' \rceil \leq p_S(\lceil input(\varphi) \rceil)$, and $\lceil \beta \rceil \leq c$ for all output labels β in φ.

A system S is *safe* when it is polynomial and, for any φ, if $S \xrightarrow{\varphi} S_1$ and $S \xrightarrow{\varphi} S_2$ then S_1 and S_2 have the same labels.

Hence, starting from a safe process, a series of labels fully determines any further observation. Safety is preserved by all transitions, and also uniformly bounds (for example) the number of local reductions, new names, and certificates.

These restrictions are serious, but they are also easily established when writing simple programs and protocols. (Still, it would be interesting to relax them, maybe using a probabilistic process calculus.) Accordingly, our language design prevents trivial sources of non-determinism and divergence (e.g. with pattern matching on values, and replicated inputs instead of full-fledged replication); further, most internal choices can be coded as external choices driven by the inputs of our abstract environment.

We can adapt usual bisimulation proof techniques to establish both equivalences and safety: instead of examining all series of labels φ, it suffices to examine single transitions for the systems in the candidate relation.

Lemma 3.2 (Bisimulation Proof). *Let $\mathcal{R}$ be a reflexive labelled bisimulation such that, for all related systems $S_1 \mathcal{R} S_2$, if $S_1 \xrightarrow{\gamma} S_1'$ and $S_2 \xrightarrow{\gamma} S_2'$, then $S_1' \mathcal{R} S_2'$.*

Polynomial systems related by $\mathcal{R}$ are safe and bisimilar.

Proof. By induction of φ, we show that $S_1 \mathcal{R} S_2$ and $S_i \xrightarrow{\varphi} S_i'$ for $i = 1, 2$ implies $S_1' \mathcal{R} S_2'$. □

Equivalences with Message Authentication; Strong Secrecy and Authentication

We illustrate our definitions using basic examples of secrecy and authentication stated as equivalences between a protocol and its specification (adapted from [AFG00]). Consider a principal a that sends a single message. In isolation, we have the equivalence $a[a{:}b\langle V\rangle] \simeq a[a{:}b\langle V'\rangle]$ if and only if $V = V'$, since the environment observes V on the label of the transition $a[a{:}b\langle V\rangle] \xrightarrow{a:b\langle V\rangle} a[\mathbf{0}]$. Consider now the system

$$S(V,W) = a[a{:}b\langle V,W\rangle] \,|\, b[(a{:}\langle ?x, \_\rangle).P],$$

with an explicit process for principal b that receives a's message and, assuming the message is a pair, runs P with the first element of the pair substituted for x. For any terms W_1 and W_2, we have $S(V,W_1) \simeq S(V,W_2)$. This equivalence states the strong secrecy of W, since its value cannot affect the environment. The system has two transitions

$$S(V,W) \xrightarrow{\nu i.a:b}\xrightarrow{(i)} a[\mathbf{0}] \,|\, b[P\{V/x\}]$$

interleaved with inputs from any $e \in \mathsf{Prin} \setminus \{a,b\}$. Further, the equivalence

$$S(V,W) \simeq a[a{:}b\langle\rangle] \,|\, b[(a{:}\langle \_\rangle).P\{V/x\}]$$

captures both the authentication of V and the absence of observable information on V and W in the communicated message, since the protocol $S(V,W)$ behaves just like another protocol that sends a dummy message instead of V, W.

Equivalences with Certificates

Let $\Phi = \{a\{m\}\}$—that is, assume a has issued a single certificate. We have

$$\begin{aligned}
\Phi \vdash a[(e{:}\langle a\{n\}\rangle).P] &\simeq \Phi \vdash a[] && (3.2)\\
\Phi \vdash a[a{:}b\langle a\{n\}\rangle \,|(e{:}\langle a\{n\}\rangle).P] \,|\, b[] &\simeq \Phi \vdash a[a{:}b\langle 0\rangle] \,|\, b[] && (3.3)\\
\Phi \vdash a[(e{:}\langle a\{x\}\rangle).P] &\simeq \Phi \vdash a[(e{:}\langle a\{\_\}\rangle).P\{m/x\}] && (3.4)
\end{aligned}$$

These three equations rely on the impossibility for the adversary to forge any certificate from a with another content. Similar equations also hold if the input is performed by another principal (as long as a does not issue any other certificate), and even if the attacker can choose arbitrary values V and W instead of the names m and n, as long as $V \neq W$. Conversely, consider the system

$$S[\_] = \Phi \vdash a[(e{:}\langle a\{m\}\ \mathit{as}\ \mathit{sig}, a\{m\}\ \mathit{as}\ \mathit{sig}'\rangle).\mathtt{match}\ \mathit{sig}\ \mathtt{with}\ \mathit{sig}'\ \mathtt{in}\ \mathbf{0}\ \mathtt{else}\ [\_]]$$

Since signatures are malleable, the else branch is reachable. Take as an example, an input labelled $(e{:}a\langle a\{m\}_0, a\{m\}_1\rangle$, hence in general $S[P] \not\simeq S[Q]$.

3.4 Applications

We present three coding examples within our language, dealing with anonymous forwarders, electronic contracts, and system initialisation. In addition, we coded a translation from asynchronous pi calculus processes into local processes, using terms $\mathtt{chan}(n)$ to represent channels. (The scope of name n represents the scope of the channel, and channel-based communications is implemented by pattern matching on channel terms.) We also coded distributed communications for the authenticated join-calculus channels of [AFG00], using certificates $a\{\mathtt{chan}(n)\}$ to represent output capabilities of channels.

3.4.1 Anonymous Forwarders

We consider a (simplified, synchronous) anonymising mix hosted by principal c. This principal receives a single message V from every participant $a \in A$, then forwards all those messages to some sender-designated address b. The forwarded message does not echo the sender identity—however this identity may be included as a certificate in the message V. We study a single round, and assume that, for this round, the participants trust c but do not trust one another. We use the following local processes (indexed by principal) and systems:

$$\begin{aligned}
P_c &= \textstyle\prod_{a\in A}(a{:}c\langle ?b, ?V\rangle).(\mathtt{tick}\,|(\mathtt{go}).c{:}b\langle \mathtt{forward}(V)\rangle)\\
Q_c &= (\mathtt{tick}).^{\text{for each } a\in A} \textstyle\prod_{a\in A}\mathtt{go}\\
P_a^\sigma &= a{:}c\langle b_{a\sigma}, V_{a\sigma}\rangle \,|\, P'_a\\
S^\sigma &= c[P_c \,|\, Q_c] \,|\, \textstyle\prod_{a\in A'} a[P_a^\sigma]
\end{aligned}$$

The process P_c receives a single message from every $a \in A$, then it emits a local `tick` message and wait for a local go message. The process Q_c runs in parallel with P_c and provides synchronisation; it waits for a `tick` message for every participant, then sends go messages to trigger the forwarding of all messages.

Let $A' \subseteq A$ be a subset of participants that comply with the protocol. We set $\mathcal{H} = A' \uplus \{c\}$. Anonymity for this round may be stated as follows: no coalition of principals in $A \setminus A'$ should be able to distinguish between two systems that differ only by a permutation of the messages sent by the participants in A'. Formally, for any such permutations σ and σ', we verify the equivalence $S^{\sigma} \simeq S^{\sigma'}$. Hence, even if the environment knows all the V messages, the attacker gains no information on σ. (Conversely, the equivalence fails, due to traffic analysis, if we use instead a naive mix that does not wait for all messages before forwarding, or that accepts messages from any sender.)

3.4.2 Electronic Payment Protocol

As a benchmark for our framework, we consider the electronic payment protocol presented by Backes and Dürmuth [BD05] that is a simplified version of the 3KP payment system [BGH$^+$95, BGH$^+$00]. We refer to their work for a detailed presentation of the protocol and its properties. The authors provide a computationally sound implementation of the protocol on top of an idealised cryptographic library [BPW03]. We obtain essentially the same security properties, but our coding of the protocol is more abstract and shorter than theirs (by a factor of 10) and yields simpler proofs, essentially because it does not have to deal with the details of signatures, marshalling, and local state—coded once and for all as part of our language implementation.

We adapt their notations, e.g. $d, p \mapsto t$. Our calculus is more abstract and formally convenient, but less expressive than machines running on top of their library. Arguably, our low-level machine description factors out (and clarifies) most of their coding on top of the library.

The protocol has four roles, a client c, a vendor v, an acquirer ac, and a trusted third party ttp. For simplicity, we assume that ac and ttp are unique and well-known. In addition, we use a distinct, abstract principal U that sends or receives all events considered in trace properties. Initially, the client, vendor, and acquirer tentatively agree on their respective identities and a (unique) transaction descriptor t that describes the goods and their price. The protocol essentially relies on the forwarding of certificates. We let $x:\{y, V\}$ abbreviates a message with a certified content $x{:}y\langle x\{y, V\}\rangle$, and use $as\ sig$ to bind the corresponding certificate $x\{y, V\}$.

A system S consists of any number of principals (potentially) running the three roles, plus a unique principal ttp running $P_{\mathtt{ttp}}$. The system should not define U, which represents an arbitrary, abstract environment that controls the actions of the other principals. For a given normal trace φ, we say that the payment $t, c, v,$ ac is *complete* when φ includes the following input labels:

(i) if $c \in \mathcal{H}$, then $U{:}c\langle \mathtt{pay}(t, v)\rangle$;

(ii) if $v \in \mathcal{H}$, then $U{:}v\langle \mathtt{receive}(t, c)\rangle$; and

(iii) if ac $\in \mathcal{H}$, then $U{:}\mathtt{ac}\langle \mathtt{allow}(t, c, v)\rangle$.

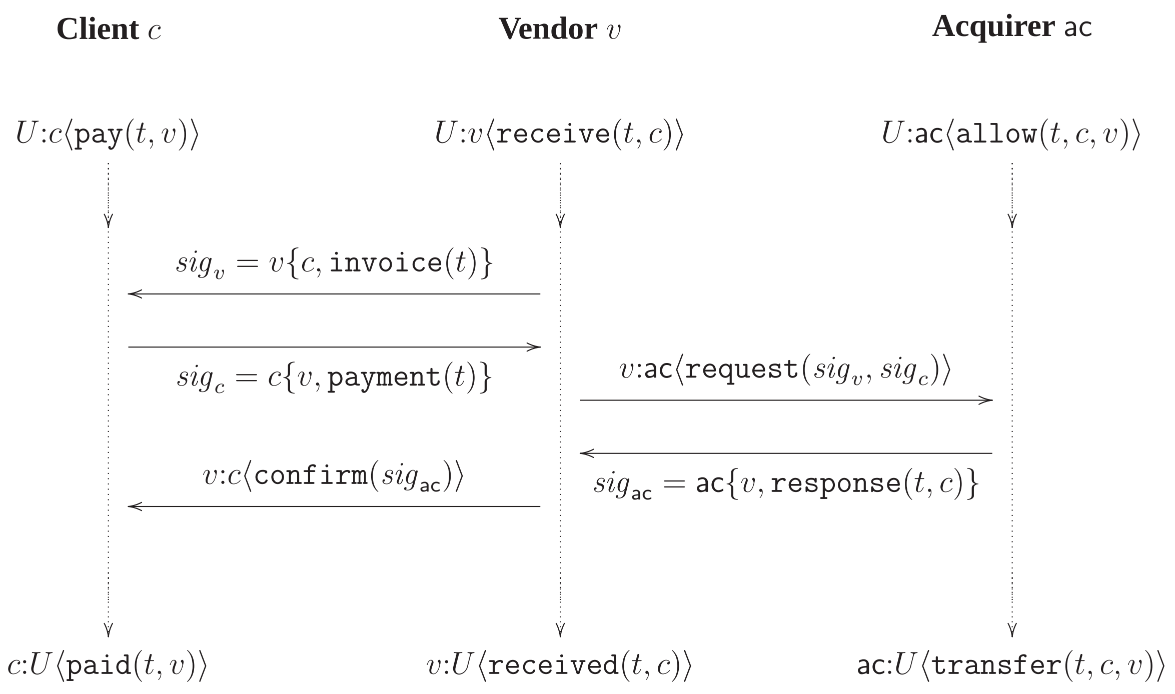
Client c
Vendor v
Acquirer ac
$U{:}c\langle\texttt{pay}(t,v)\rangle$
$U{:}v\langle\texttt{receive}(t,c)\rangle$
$U{:}\texttt{ac}\langle\texttt{allow}(t,c,v)\rangle$
$sig_v = v\{c, \texttt{invoice}(t)\}$
$sig_c = c\{v, \texttt{payment}(t)\}$
$v{:}\texttt{ac}\langle\texttt{request}(sig_v, sig_c)\rangle$
$sig_{\texttt{ac}} = \texttt{ac}\{v, \texttt{response}(t,c)\}$
$v{:}c\langle\texttt{confirm}(sig_{\texttt{ac}})\rangle$
$c{:}U\langle\texttt{paid}(t,v)\rangle$
$v{:}U\langle\texttt{received}(t,c)\rangle$
$\texttt{ac}{:}U\langle\texttt{transfer}(t,c,v)\rangle$

Figure 3.1: Diagram of the Electronic Payment Protocol [BD05]

We can now state the following properties:

- *Weak atomicity* is a trace property expressed as follows: if φ includes any output of the form $c{:}U\langle\texttt{paid}(t,v)\rangle$, $v{:}U\langle\texttt{received}(t,c)\rangle$, or $\texttt{ac}{:}U\langle\texttt{transfer}(t,c,v)\rangle$, then the payment $t, c, v,$ ac is complete.
- *Correct client dispute* states that an honest client—who starts a dispute for transaction t only after completing the protocol for t, as coded in the last line of *Client$_c$*—always wins his dispute: that is, for any trace φ, if $c \in \mathcal{H}$ and $c{:}U\langle\texttt{paid}(t,v)\rangle$ is in φ, then $\texttt{ttp}{:}U\langle\texttt{reject}(t,c,v)\rangle$ is not in φ. (This property is rather weak, as the vendor and acquirer complete the protocol before the client.)
- *Correct vendor dispute* and *Correct acquirer dispute* are similar to the previous property and we omit it here.
- *No framing* states that the ttp does not wrongly involve parties that have not initiated the protocol with matching parameters. It is a variant of weak atomicity: outputs of the form $\texttt{ttp}{:}U\langle\texttt{accept}(t,c,v)\rangle$ only occur for complete payments.

These properties are directly established by induction on the high-level transitions of S.

Sketch of the Proof. By induction on the trace φ. We show that the state of the system is determined by φ, and that every enabled input in this state yields outputs that meet the claimed

$$\begin{array}{rcl}
\mathit{Client}_c & = & *(U{:}c\langle \mathtt{pay}(?t,?v)\rangle).\\
 & & \quad (v:\{c,\mathtt{invoice}(t)\}\ \mathit{as}\ \mathit{sig}_v).\\
 & & \qquad (c:\{v,\mathtt{payment}(t)\}\mid\\
 & & \qquad (v{:}c\langle \mathtt{confirm}(\mathsf{ac}\{v,\mathtt{response}(t,c)\}\ \mathit{as}\ \mathit{sig}_{\mathsf{ac}})\rangle).\\
 & & \qquad\quad (c{:}U\langle \mathtt{paid}(t,v)\rangle\mid\\
 & & \qquad\quad (U{:}c\langle \mathtt{dispute}(t)\rangle).c{:}\mathsf{ttp}\langle \mathtt{client\_dispute}(\mathit{sig}_v,\mathit{sig}_{\mathsf{ac}})\rangle)))\\
\mathit{Vendor}_v & = & *(U{:}v\langle \mathtt{receive}(?t,?c)\rangle).\\
 & & \quad (v:\{c,\mathtt{invoice}(t)\}\ \mathit{as}\ \mathit{sig}_v\mid\\
 & & \quad (c:\{v,\mathtt{payment}(t)\}\ \mathit{as}\ \mathit{sig}_c).\\
 & & \qquad (v{:}\mathsf{ac}\langle \mathtt{request}(\mathit{sig}_v,\mathit{sig}_c)\rangle\mid\\
 & & \qquad (\mathsf{ac}:\{v,\mathtt{response}(t,c)\}\ \mathit{as}\ \mathit{sig}_{\mathsf{ac}}).\\
 & & \qquad\quad (v{:}c\langle \mathtt{confirm}(\mathit{sig}_{\mathsf{ac}})\rangle\mid v{:}U\langle \mathtt{received}(t,c)\rangle\mid\\
 & & \qquad\quad (U{:}v\langle \mathtt{dispute}(t)\rangle).v{:}\mathsf{ttp}\langle \mathtt{vendor\_dispute}(\mathit{sig}_c,\mathit{sig}_{\mathsf{ac}})\rangle))))\\
\mathit{Acquirer}_{\mathsf{ac}} & = & *(U{:}\mathsf{ac}\langle \mathtt{allow}(?t,?c,?v)\rangle).\\
 & & \quad (v{:}\mathsf{ac}\langle \mathtt{request}(v\{c,\mathtt{invoice}(t)\}\ \mathit{as}\ \mathit{sig}_v,\\
 & & \qquad\qquad\qquad c\{v,\mathtt{payment}(t)\}\ \mathit{as}\ \mathit{sig}_c)\rangle).\\
 & & \qquad (\mathsf{ac}:\{v,\mathtt{response}(t,c)\}\mid \mathsf{ac}{:}U\langle \mathtt{transfer}(t,c,v)\rangle\mid\\
 & & \qquad (U{:}\mathsf{ac}\langle \mathtt{dispute}(t)\rangle).\mathsf{ac}{:}\mathsf{ttp}\langle \mathtt{acquirer\_dispute}(\mathit{sig}_c,\mathit{sig}_v)\rangle))\\
P_{\mathsf{ttp}} & = & *(?c{:}\mathsf{ttp}\langle \mathtt{client\_dispute}(?d)\rangle).\\
 & & \quad \mathtt{match}\ d\ \mathtt{with}\ ?v\{c,\mathtt{invoice}(?t)\},\mathsf{ac}\{v,\mathtt{response}(t,c)\}\ \mathtt{in}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{accept\_client}(t,c,v)\rangle\ \mathtt{else}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{reject\_client}(t,c,v)\rangle\mid\\
 & & *(?v{:}\mathsf{ttp}\langle \mathtt{vendor\_dispute}(?d)\rangle).\\
 & & \quad \mathtt{match}\ d\ \mathtt{with}\ ?c\{v,\mathtt{payment}(?t)\},\mathsf{ac}\{v,\mathtt{response}(t,c)\}\ \mathtt{in}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{accept\_vendor}(t,c,v)\rangle\ \mathtt{else}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{reject\_vendor}(t,c,v)\rangle\mid\\
 & & *(\mathsf{ac}{:}\mathsf{ttp}\langle \mathtt{acquirer\_dispute}(?d)\rangle).\\
 & & \quad \mathtt{match}\ d\ \mathtt{with}\ ?c\{?v,\mathtt{payment}(?t)\},v\{c,\mathtt{invoice}(t)\}\ \mathtt{in}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{accept\_acquirer}(t,c,v)\rangle\ \mathtt{else}\\
 & & \qquad \mathsf{ttp}{:}U\langle \mathtt{reject\_acquirer}(t,c,v)\rangle
\end{array}$$

Figure 3.2: Encoding of the Electronic Payment Protocol [BD05]

properties. (In contrast with [BD05], we don't have to define complex, auxiliary invariants; the invariant directly follows from our definition of labelled transitions.) □

3.4.3 Initialisation

This technical example shows that, without loss of generality, it suffices to develop concrete implementations for *initial systems* that do not share any names, certificates, or intercepted messages between principals and the environment. Up to structural equivalence, every system is of the form $S = \Phi \vdash \nu\widetilde{n}.(\prod_{a\in\mathcal{H}} a[P_a] \mid \prod_{i\in I} M/i)$. The sharing of names and certificates between principals and the environment can be quite complex, and is best handled using an ad hoc (but high-level) "bootstrapping" protocol, outlined below:

1. Free names of S and restricted non-local names $\widetilde{n}$ are partitioned between honest principals; let $(n_{a,1},\dots,n_{a,k_a})_{a\in\mathcal{H}}$ be those names.

2. Free names and non-self-issued certificates that occur in the local processes P_a are exchanged using a series of initialisation messages $M_{ab,r}$ of the form

   $$M_{ab,r} = a{:}b\langle \mathtt{init}_{ab,r}(n_{a,1},\dots,n_{a,k_{a_r}}, a\{V_{ab,1}\},\dots,a\{V_{ab,m_r}\})\rangle,$$

   carrying names and certificates issued by a that occur in P_b. Similarly, initialization messages sent to a fixed principal $e \notin \mathcal{H}$ export the free names of S and the certificates of Φ, whereas initialization messages from e import certificates issued by principals not in $\mathcal{H}$.

   Each principal $a \in \mathcal{H}$ thus sends a series of initialisation messages, and sequentially receives and checks all initialisation messages addressed to him, using input patterns of the form $(T_{ba,r})$ where $T_{ba,r}$ is $M_{ba,r}$ with binding variables $?n_1,\dots,?n_k$ instead of the names and aliases $b\{V_{ba,r}\}\ as\ ?x$ for checking and binding certificates. The whole local initialisation process is guarded by a dummy input with pattern $T_{ea,0} = e{:}a\langle \_ \rangle$, so that the initial system be stable.

3. Finally, each principal a sends a message M for every intercepted message M/i from a defined in S, then starts P_a.

For instance, in case $\mathcal{H} = \{a, b\}$ with neither nested certificates nor intercepted messages, the local initialisation process for a is

$$P_a^\circ = (T_{ea,0}).\nu n_1,\dots,n_{k_a}.(M_{ab,1} \mid M_{ae,1} \mid (T_{ba,1}).(T_{ea,1}).P_a)$$

In the general case, several rounds of initialisation messages may be needed to exchange certificates whose contents include names and certificates, and to emit messages with the same shape one at a time.

Intuitively, the attacker may prevent P_a from running at all by not forwarding messages, or provide a message whose certificates do not match the certificates expected by P_a, but it could block all of a's communications anyway. If P_a does start, it does so with the right names and certificates.

The next lemma states the correctness of the initialisation protocol. The second property of the lemma states that an environment that follows the protocol always reaches S_i.

Lemma 3.3 (Initialisation). *Let S_i for $i = 0, 1$ be safe stable systems with the same principals, exported certificates, and intercepted-message labels.*

There exist initial safe stable systems S_i° and labels φ° such that

1. *we have normal transitions $S_i^\circ \xrightarrow{\varphi^\circ} S_i$;*
2. *any normal transitions $S_i^\circ \xrightarrow{\varphi^\circ} S'$ imply that $S' \equiv S_i$; and*
3. *$S_0 \simeq S_1$ if and only if $S_0^\circ \simeq S_1^\circ$.*

Proof. (Sketch.) We have $S_i^\circ \xrightarrow{\varphi^\circ} S_i$ deterministically, so $S_0^\circ \simeq S_1^\circ$ implies $S_0 \simeq S_1$. Conversely, we show that the relation

$$\begin{aligned}\mathcal{R} = \{S_0', S_1') \text{ such that } S_0 \simeq S_1, S_i^\circ \xrightarrow{\varphi} S_i',& \\ \text{and } \varphi \text{ is a prefix of a permutation of the labels of } \varphi^\circ\} \cup \simeq&\end{aligned}$$

is a labelled bisimulation. (Intuitively, φ is the part of φ° that has already been enabled by the attacker.) □

3.5 A Concrete Implementation

We are now ready to define the machines outlined in Section 3.1, relying on translations from high-level terms and processes to keep track of their runtime state. We systematically map high-level systems S to the machines of Section 3.1, mapping each principal $a[P_a]$ of S to a PPT machine M_a that executes P_a. We start by giving an outline of our implementation.

The implementation mechanisms are simple, but they need to be carefully specified and composed. (As a non-trivial example, when a machine outputs several messages, possibly to the same principals, we must sort the messages after encryption so that their ordering on the wire leaks no information on the computation that produced them.)

We use two concrete representations for terms: a wire format for (signed, encrypted) messages between principals, and an internal representation for local terms. Various bitstrings represent constructors, principal identities, identifiers for names, and certificates. Marshaling and unmarshaling functions convert between internal and wire representations. When marshaling a locally restricted name identifier ind for the first time, we draw a bitstring s of length η uniformly at random, associate it with ind, and use it to represent ind on the wire. When unmarshaling a bitstring s into an identifier for a name, if s is not associated with any local identifier, we create a new internal identifier ind for the name, and also associate s with ind.

Signatures are verified as part of unmarshaling. Signatures for self-issued certificates are generated on-demand, as part of marshaling, and cached, so that the same signature value is used for any certificate with identical content.

Local processes are represented in normal form for structural equivalence, using internal terms and multisets of local inputs, local outputs, and outgoing messages. We implement reductions using an abstract machine that matches inputs and outputs using an arbitrary deterministic, polynomial-time scheduler.

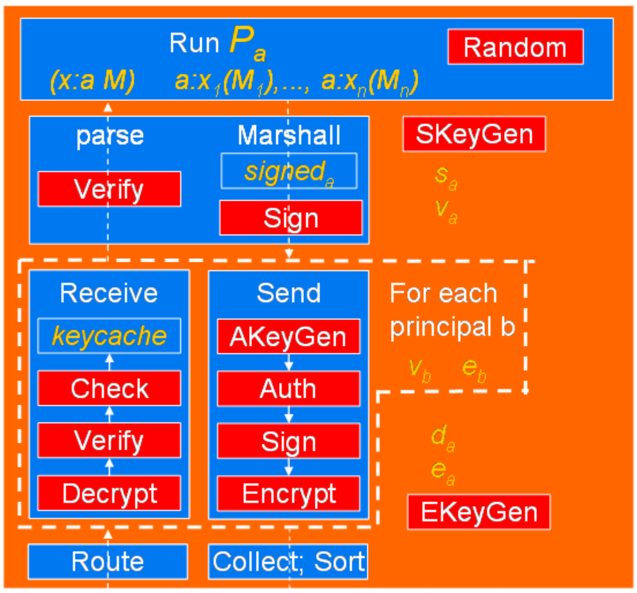

Figure 3.3: Local machine for principal a connected to the adversary machine

3.5.1 Implementation of Machines

The transition rules of Section 3.2.2 declare that all communications be authentic and confidential. In order to meet these requirements, our implementation relies on concrete bitstrings and cryptographic protocols.

Definition 3.18 (Low-Level State). The runtime state of machine M_a consists of the following data:

- id_a, d_a, and s_a are bitstrings that represent the low-level identifier for principal a and its private keys for decryption and signing.
- $peers = \{(id_u, e_u, v_u) \mid u \in \mathsf{Prin}\}$ binds, for every principal, a low-level identifier to public keys for encryption and signature verification.
- p_a is a low-level representation of a local process running at a (defined below).
- $keycache_a$ is a set of authentication keys for all received messages.
- $signed_a$ is a partial function from certificates issued by a to signature values.
- $names_a$ is a partial function from name identifiers to bitstrings.

The main machine components are depicted in Figure 3.3.

Before detailing the definitions of all the protocols presented if Figure 3.3, we describe a complete run of the machine. Recall that M_a is connected to the environment by two wires, $\mathbf{in}_a$ and $\mathbf{out}_a$. The wire format for messages is the concatenated bitstring $id_u\_id_v\_msg$ where u and v are the (apparent) sender and receivers and msg is some encrypted, authenticated, marshaled message. When it receives such a message (with $id_v = id_a$), M_a uses id_u to dispatch msg to the `receive` protocol (Definition 3.23) for remote principal u— there is an instance of the `receive` protocol for each peer principal u. The protocol verifies the freshness, integrity, and authenticity of the message, updates $keycache_a$, then returns a decrypted bitstring s. If a verification step fails, the message is discarded.

At this stage, msg is a genuine message from u to a, but its content is not necessarily well-formed. For instance u may have included a certificate apparently issued by b but with an invalid signature. Content validation occurs as s is unmarshaled (Definition 3.21) from its wire format into some internal (trusted) representation $parse_a(s)$ of a high-level term V. In particular, this trusted representation embeds a valid signature for every certificate of V. After successful reception and unmarshaling, a representation m of the incoming message $u{:}a\langle V\rangle$ may react with an input within p_a and trigger local computations. To this end, a local interpreter (Definition 3.19) derived from the abstract machine of Section 3.2.3 runs on $p_a \mid m$. If the interpreter terminates, it yields a new stable internal process p'_a plus a set of outgoing messages X to be sent to the network.

Each message $a{:}u_i\langle V_i\rangle$ represented in X is then marshaled (Definition 3.20) and passed to the instance of the `send` protocol (Definition 3.22) associated with the intended recipient u_i. The resulting bitstrings, all in wire format, of the form $id_a\_id_{u_i}\_msg_i$, are eventually sorted (by receiving principal, then encrypted value msg_i)—to ensure that their ordering leaks no information on their payload or their internal production process—and written on $\mathbf{out}_a$. A final `done` bitstring is issued and the machine terminates. (Hence, for instance, if p does not react with m, the machine simply writes done on $\mathbf{out}_a$ and terminates.)

Next, we describe in turn each of the components of the local machine.

Low-level Processes Reductions

The internal representation of terms uses the same grammar as in the high-level language except for atomic subterms: principals u are boxed, fixed-sized bitstrings $\mathtt{prin}(id_u)$ ($\ell_{\mathtt{prin}}$); free names are boxed, bitstrings $\mathtt{name}(ind)$ where ind is an internal identifier for names; and certificate labels are linear-sized bitstrings s such that either s is a valid signature for the certificate or $s = \mathtt{0}$ and the certificate is self-issued. Bound variables and names may still occur in terms under input guards.

Definition 3.19 (Internal Reductions). The local reduction algorithm refines the abstract machine of Section 3.2 as follows:

1. it represents the multisets X, M, and G using internal terms;

2. it uses a deterministic, polynomial-time, complete scheduler;

3. instead of lifting new name restriction $\nu n.Q$, it generates a new identifier ind (possibly incrementing an internal counter) and substitutes $\mathtt{name}(ind)$ for all bound instances of the n in Q.

Marshaling and Unmarshaling Protocols

These algorithms are responsible for processing messages that are about to be sent to (that were received from) the network. The marshaling process transforms each internal term into a bitstring to be sent over the network, and the unmarshal algorithm attempts to transform a bitstring received from the network to a (trusted) internal term; it may instead return an error if the message is not well-formed, or if the signature of an included certificate cannot be verified. In any of these cases the entire message is discarded.

We use a fixed, injective function from all constructors plus `name` and `prin` to bitstrings of a given fixed size; we still write `f`, `name`, `prin` for the corresponding bitstrings. We write s_s' for the bitstring obtained by concatenating s and s'.

Definition 3.20 (Marshaling). Let $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a signature scheme. The function $\llbracket \cdot \rrbracket$ maps principal's internal representations of closed terms to bitstrings, as follows:

$$\begin{array}{rcll}
\llbracket \mathtt{name}(ind) \rrbracket & = & \mathtt{name}\_names(ind) & \\
 & & \text{adding } names(ind) = s \longleftarrow \{0,1\}^{\eta} \text{ when undefined} & \\
\llbracket \mathtt{prin}(s) \rrbracket & = & \mathtt{prin}\_s & \\
\llbracket \mathtt{f}(v_1, \dots, v_n) \rrbracket & = & \mathtt{f}\_\llbracket v_1 \rrbracket\_\dots\_\llbracket v_n \rrbracket & \text{when } \mathtt{f} \notin \{\mathtt{name}, \mathtt{prin}, \mathtt{cert}\} \\
\llbracket v_1\{v_2\}_s \rrbracket & = & \mathtt{cert}\_\llbracket v_1 \rrbracket\_\llbracket v_2 \rrbracket\_s & \text{when } s \neq \mathsf{0} \\
\llbracket v_1\{v_2\}_{\mathsf{0}} \rrbracket & = & \mathtt{cert}\_\llbracket v_1 \rrbracket\_\llbracket v_2 \rrbracket\_signed(v_1\{v_2\}_{\mathsf{0}}) & \text{when } v_1 = \mathtt{prin}(id_b), b \in \mathcal{H} \\
 & & \text{adding } signed(v_1\{v_2\}_{\mathsf{0}}) = \mathcal{S}(s_b, \llbracket v_2 \rrbracket_b) \text{ when undefined} &
\end{array}$$

We denote by $\llbracket \cdot \rrbracket_a$ the marshaling procedure for machine M_a that uses only uses tables $names_a$ and $signed_a$, that is, $names = names_a$ and $signed = signed_a$.

We prove as an invariant that for all certificates of the form $v_1\{v_2\}_{\mathsf{0}}$ in p_a, $v_1 = \mathtt{prin}(id_a)$, hence $\llbracket \cdot \rrbracket_a$ is defined for all internal representations of terms of M_a.

We assume that after marshalling, and before sending, all our messages are padded to a fixed length that is given by the polynomial $ms(\eta)$, a parameter of the implementation. We could have assumed that this was not the case and if so, we needed to consider this difference of length in our high-level semantics. We could have done it using sorts and sizes for input and output messages.

Definition 3.21 (Unmarshaling). Let $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a signature scheme. The partial function $parse(\cdot)$ maps bitstrings to internal representations of closed terms, as follows, and fails in all other cases.

$$\begin{array}{rll}
parse(\texttt{name\_}s) = \texttt{name}(ind) & \text{when, there is } ind : names(ind) = s \\
& \text{otherwise if } |s| = \eta \text{ then} \\
& \quad ind = |dom(names)| + 1 \text{ and} \\
& \quad \text{add } names(ind) = s \\
parse(\texttt{prin\_}s) = \texttt{prin}(s) & \text{when } |s| = \ell_{\texttt{prin}} \text{ and } (s, e_s, v_s) \in peers \\
parse(\texttt{f\_}s_1\texttt{\_} \dots s_n) = \texttt{f}(v_1, \dots, v_n) & \text{when } \texttt{f} \notin \{\texttt{name}, \texttt{prin}, \texttt{cert}\} \text{ has arity } n \\
& parse(s_i) = v_i \text{ for } i = 1..n \\
parse(\texttt{cert\_}s_1\texttt{\_}s_2\texttt{\_}s_3) = v_1\{v_2\}_s & \text{when, for some } (id_u, e_u, v_u) \in peers, \\
& parse(s_1) = \texttt{prin}(id_u) = v_1, \\
& parse(s_2) = v_2 \\
& \mathcal{V}(v_u, s_2, s_3) = 1 \\
& s = \text{if } signed(v_1\{v_2\}_{\texttt{0}}) = s_3 \text{ then } \texttt{0} \text{ else } s_3
\end{array}$$

We denote by $parse_a(\cdot)$ the unmarshaling procedure for machine M_a that uses only uses tables $names_a$ and $signed_a$, that is, $names = names_a$ and $signed = signed_a$.

Unmarshaling includes signature verification for any received certificate, and is otherwise standard; it is specified here as a partial function from strings to internal representations, and can easily be implemented as a parser. Our treatment of self-issued certificates with label `0` reflects our choice of internal representations: `0` stands for the (unique) signature generated by the local machine for this certificate content, the first time this certificate is marshaled. (In addition, the adversary may be able to derive a variant of this certificate with a different signature, unmarshaled with a non-zero label; such certificates are then treated using the default case for marshaling.)

Although we give a concrete definition of $[\![\cdot]\!]$, $parse(\cdot)$, and message formats, our results only depend on their generic properties. We only require that, for a given local machine, every string be unmarshaled to at most one internal term, whose marshaling yields back the original string, that is, $parse_a([\![\ulcorner V \urcorner^a]\!]_a) = \ulcorner V \urcorner^a$. ($\ulcorner V \urcorner^a$ denotes the internal representation of a for V. We define $\ulcorner V \urcorner^a$ formally in Definition 3.25.) For simplicity, we have that the length of the string be a function of the structure of the internal term and of the security parameter.

Sending and Receiving Protocols

Two important pieces of our systems are the $\texttt{send}_b$ and $\texttt{receive}_b$ protocols. There are one pair of these for each other principal. The `send` protocol defined below, ensures that, as abstracted in the high-level semantics, all communications are opaque for the adversary using public-key encryption, and that the communication is authentic, using authentication and signature schemes. This protocol takes a bitstring s (containing a marshaled message from a to u), protects it, and returns it in wire format. Conversely, the receiving protocol takes a message in wire format presumably from u, verifies it, and returns its payload. We also request robustness against replay attacks; after decryption, we reject any message whose authentication key is already recorded.

These protocols are intended as a simple example; other choices are possible. We may for instance consider long term shared keys between principals, in order to reduce the overhead of public-key cryptography. If we decide to do so, we should introduce a nonce in the message that is encrypted in Step 3.

Definition 3.22 (Sending to u). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, and $\Lambda = (\mathcal{G}_\Lambda, \mathcal{A}, \mathcal{C})$ be respectively an encryption, signature, and authentication schemes. Given a bitstring s, the $\mathtt{send}_u$ protocol

1. generates a fresh authentication key $k \longleftarrow \mathcal{G}_\Lambda(1^\eta)$;
2. computes $m = s\_id_a\_k\_\mathcal{S}(s_a, k\_id_u)\_\mathcal{A}(k, s)$;
3. computes $msg = \mathcal{E}(e_u, m)$; and
4. returns $id_a\_id_u\_msg$.

Definition 3.23 (Receiving from u). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, and $\Lambda = (\mathcal{G}_\Lambda, \mathcal{A}, \mathcal{C})$ be respectively an encryption, signature, and authentication schemes. Given a bitstring $id_u\_id_a\_msg$, the $\mathtt{receive}_u$ protocol

1. computes $s\_id_u\_k\_s_{sig}\_s_{auth} = \mathcal{D}(d_a, msg)$;
2. checks that there is an entry $(id_u, e_u, v_u) \in peers$ with $\mathcal{V}(v_u, k\_id_a, s_{sig}) = 1$;
3. checks that $\mathcal{C}(k, s, s_{auth}) = 1$;
4. checks that k is not in $keycache$, and adds it to $keycache$;
5. returns s.

The entire message is discarded if any step of the protocol fails.

Mapping High-Level Systems to Low-Level Machines

In order to systematically relate the runtime state of low-level machines to the abstract state of high-level systems, we define an associated *shadow state*. This structure provides a consistent interpretation of terms across machines. In combination, a system and its shadow state determine their implementation, obtained as a compositional translation of terms, local processes, and configurations. (This state is shadow as it need not be maintained at runtime in the low-level implementation; it is used solely as an abstraction to reason about the correctness of our implementations.) We further partition this state into public parts, intended to be part of the attacker's knowledge, and private parts.

Definition 3.24 (Shadow State). Let $S = \Phi \vdash \nu\tilde{n}.C$ be a system such that the configuration $C = \prod_{a \in \mathcal{H}} a[P_a] \mid \prod_{i \in I} M/i$ is in normal form. A *shadow state* for S, written D, consists of the following data structure:

- $prin \in \mathsf{Prin} \to (\{0,1\}^\eta)^5$ is a function from $u \in \mathsf{Prin}$ to bitstrings id_u, e_u, v_u, d_u, s_u such that $u \to id_u$ is injective, and for every $u \in \mathcal{H}$, we have $(e_u, d_u) \longleftarrow \mathcal{K}(1^\eta)$, and $(v_u, s_u) \longleftarrow \mathcal{G}(1^\eta)$.

  The bitstrings id_u, e_u, v_u are public for all $u \in \mathsf{Prin}$; d_u and s_u are public if $u \in \mathsf{Prin}\backslash\mathcal{H}$.

- $name \in \mathsf{Name} \rightharpoonup \{0,1\}^\eta$ is a partial injective function defined at least on every name that occurs free in S, and names that occur in Φ, $\mathsf{D}.certval$ or $\mathsf{D}.wire$.

  The bitstring $name(m)$ is public for every name $m \notin \widetilde{n}$.

- ni is a family of partial injective functions $ni^a : \mathsf{Name} \rightharpoonup \{0,1\}^\eta$ for each $a \in \mathcal{H}$, defined at least for all names of P_a that are not locally-restricted.

- $certval$ is a partial function from certificates $u\{V\}_\ell$ to $s \in \{0,1\}^\eta$ defined at least on the certificates of Φ, $\mathsf{D}.wire$, and all certificates of P_a of the form $a\{V\}_\ell$ with $\ell \neq 0$ or $u\{V\}_\ell$ with $u \neq a$. It is also defined for all the certificates in V such that $u\{V\}_\ell$ is defined in $certval$. $certval$ satisfies the following property: if $certval(u\{V\}_\ell) = s$, then $\mathcal{V}(v_u, [\![\ulcorner V^{\urcorner \mathsf{D},u}]\!], s) = 1$.

  The bitstring $certval(V)$ is public when $V \in \mathcal{M}(\Phi)$ or V issued by $u \notin \mathcal{H}$.

- $wire$ is a partial function from indices i to (M, k, s, del) defined at least on I, where $M = a{:}b\langle V\rangle$ with $a, b \in \mathcal{H}$, and $del = 0$ if $i \in I$ and $del = 1$ otherwise. The bitstrings s and k are the output and the authentication key produced by $\mathtt{send}_b$ on input $[\![\ulcorner V^{\urcorner \mathsf{D},a}]\!]$.

  The bitstrings s and del are public.

- $keycache$ is a function from $a \in \mathcal{H}$ to sets of bitstrings such that, if there exists an i with $wire(i) = (M, k, \_, 1)$ with M to a, then $k \in keycache(a)$.

- $ms^{\mathsf{D}}(\eta)$ is a polynomial that sets the padding-size of the implementations of S.

Intuitively, $wire$ records all messages sent between honest principals; $keycache(a)$ records the authentication keys of all messages received by a so far; it contains at least the keys of messages in $wire$ that were already received by a. When D is clear from the context, we write $prin(a)$ instead of $\mathsf{D}.prin(a)$, and similarly for the other components of D. We denote by $public(\mathsf{D})$ the binary representation of the public parts of D. When we are not interested in the specific bitstrings, we call it *shape of* D.

Definition 3.25 (Concrete Terms and Processes). A shadow state D and a set of principals $X \subseteq \mathsf{Prin}$, define a partial map from high-level terms V to internal terms as follows:

- $\ulcorner n^{\urcorner \mathsf{D},X} = \begin{cases} \mathtt{name}(ind) & \text{, if } ind = ni^u(n) \text{ for all } u \in X \\ \perp & \text{, otherwise} \end{cases}$

- $\ulcorner u^{\urcorner \mathsf{D},X} = \mathtt{prin}(\pi_1(prin(u)))$ for any principal $u \in \mathsf{Prin}$;

- $\ulcorner u\{V\}_0{}^{\urcorner \mathsf{D},X} = \ulcorner u^{\urcorner \mathsf{D},X}\{\ulcorner V^{\urcorner \mathsf{D},X}\}_0$, if $u \in X$;

- $\ulcorner u\{V\}_\ell \urcorner^{\mathsf{D},X} = \ulcorner u \urcorner^{\mathsf{D},X}\{\ulcorner V \urcorner^{\mathsf{D},X}\}_s$ where $s = \mathit{certval}(u\{V\}_\ell)$ ($u \in \mathsf{Prin}$);
- $\ulcorner \mathtt{f}(V_1, \dots, V_n) \urcorner^{\mathsf{D},X} = \mathtt{f}(\ulcorner V_1 \urcorner^{\mathsf{D},X}, \dots, \ulcorner V_n \urcorner^{\mathsf{D},X})$ for any $f \neq \mathtt{cert}$ with arity n.

We extend this map to translate local processes to low-level processes, as follows: high-level terms within local processes are translated as above, except for variables and locally-restricted names (left unchanged); high-level patterns are translated by applying the translation to all high-level terms in the pattern and leaving the rest unchanged; local processes P are translated to internal processes $\ulcorner P \urcorner^{\mathsf{D},X}$ by translating their high-level terms to internal terms.

As a corollary, we have that if D is a shadow state for S, and $a \in \mathsf{Prin}$ then $\ulcorner \cdot \urcorner^{\mathsf{D},a}$ is defined for every subterm and subprocess of S and D ($\ulcorner \cdot \urcorner^{\mathsf{D},a}$ denotes $\ulcorner \cdot \urcorner^{\mathsf{D},\{a\}}$). We often write $\ulcorner V \urcorner$ instead of $\ulcorner V \urcorner^{\mathsf{D},a}$ when D and a are clear from the context. Our intent is that, with overwhelming probability, we have $V = V'$ iff $\ulcorner V \urcorner^{\mathsf{D},a} = \ulcorner V' \urcorner^{\mathsf{D},a}$ whenever D defines these representations.

We would like to point out that the previous definition is well-formed. One should first notice that we do not translate high-level terms (hence, high-level certificates) with variables and locally-restricted names. Hence, when applying $\ulcorner u\{V\}_\ell \urcorner^{\mathsf{D},a}$, we can be sure that the certificate was previously generated and hence defined in $\mathsf{D}.\mathit{certval}$.

Definition 3.26 (System Implementations). Let S be a system with shadow state D. The implementation of S and D is the collection of machines $\mathsf{M}(S, \mathsf{D}) = (\mathsf{M}_a(S, \mathsf{D}))_{a \in \mathcal{H}}$ where each machine $\mathsf{M}_a(S, \mathsf{D})$ has the following state:

- $id_a, d_a, s_a, \mathit{peers}_a$ are read from $\mathsf{D}.\mathit{prin}$;
- $p_a = \ulcorner P_a \urcorner^{\mathsf{D},a}$;
- $\mathit{keycache}_a = \mathit{keycache}(a)$;
- $\mathit{signed}_a(\ulcorner a\{V\}_{\mathsf{o}} \urcorner^{\mathsf{D},a}) = \mathit{certval}(a\{V\}_{\mathsf{o}})$ when defined;
- $\mathit{names}_a(ni^a(n)) = \mathit{name}(n)$ when defined,

and uses $[\![\cdot]\!]_a$ and $\mathit{parse}_a(\cdot)$ as the marshaling and unmarshaling algorithms, and $ms^{\mathsf{D}}(\cdot)$ as the padding size.

3.6 Main Results

In this section we present the main results of this Chapter. Throughout this section we assume that the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is CCA-2 secure (recalled in Definition A.4), and the signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ and authentication scheme $\Lambda = (\mathcal{G}_\Lambda, \mathcal{A}, \mathcal{C})$ are CMA-secure (recalled in Definition A.6).

Our main theorems are stated in terms of arbitrary systems S. As it is convenient to have a formulation of these theorems in terms of arbitrary systems, one should not forget that an arbitrary system S is obtained starting from an initial system S° that has no shared names or certificates and no intercepted messages so, whenever we refer to a system S, we are in fact referring to its initial state S° plus its initialisation procedure. The same happens with the implementations and for that we introduce the notion of *valid shadow*. Intuitively, a shadow D is a valid shadow for S, if there is an interactive run (Definition 3.1) that starts with $\mathsf{M}(S^\circ, \mathsf{D}^\circ)$ and leads the machine to state $\mathsf{M}(S, \mathsf{D})$, where D° is the shadow obtained from D by erasing everything except $\mathsf{D}.prin$. D° is called an *initial shadow* for S. We denote by $\mathsf{A}_\circ[\mathsf{M}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{M}(S, \mathsf{D}))$ such run, where $s_{\mathtt{r}}$ is the bitstring returned by $\mathsf{A}_\circ$ at the end of the run.

Accordingly, we *define* a low level run starting from S with (valid) shadow D against A, written $\mathsf{A}[\mathsf{M}(S, \mathsf{D})] \longrightarrow s_{\mathtt{r}}(\mathsf{M})$, as $(\mathsf{A}_\circ; \mathsf{A})[\mathsf{M}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{M})$ where $(\mathsf{A}_\circ; \mathsf{A})$ represents an adversary that first runs $\mathsf{A}_\circ$ and then runs A.

Definition 3.27 (Valid Shadow). Let S be a safe system with shadow D. We say that D is a *valid shadow* for S if there exist an initial safe system S° with initial shadow D°, normal transitions $S^\circ \xrightarrow{\varphi^\circ} S$, and a PPT algorithm $\mathsf{A}_\circ$ such that $\mathsf{A}_\circ[\mathsf{M}(S^\circ, \mathsf{D}^\circ)] \longrightarrow public(\mathsf{D})(\mathsf{M}(S, \mathsf{D}))$, and $ms^{\mathsf{D}}(\eta) \geq \max_{\lceil M \rceil \leq c} \lceil [\![\ulcorner M \urcorner^{\mathsf{D}}]\!] \rceil$ where c is the constant given by the safety condition and $[\![\ulcorner M \urcorner^{\mathsf{D}}]\!]$ is the result of marshaling the low-level representations of M.

We say that D is a valid shadow for two safe systems $S_1 \simeq S_2$, if the same $\mathsf{A}_\circ$ initialises both $\mathsf{M}(S_1, \mathsf{D})$ and $\mathsf{M}(S_2, \mathsf{D})$, and $ms^{\mathsf{D}}(\eta) \geq \max_{\lceil M \rceil \leq \max\{c_1, c_2\}} \lceil [\![\ulcorner M \urcorner^{\mathsf{D}}]\!] \rceil$, where c_1 and c_2 are the constants given by the safety condition of S_1 and S_2 respectively.

Our first theorem expresses the completeness of our high-level transitions: every low-level attack can be described in terms of high-level transitions. More precisely, the probability that an interaction with a PPT adversary yields a machine state unexplained by any high-level transitions is negligible.

Theorem 3.4 (Completeness for Reachable States). *Let S be a safe stable system, D a valid shadow for S, and A a PPT algorithm.*

The probability that $\mathsf{A}[\mathsf{M}(S, \mathsf{D})]$ completes and leaves the system in state M' with $\mathsf{M}' \neq \mathsf{M}(S', \mathsf{D}')$ for any normal transitions $S \xrightarrow{\varphi} S'$ with valid shadow D' is negligible.

Proof Sketch. We just sketch the proof and refer the reader to Appendix C for the full constructions and proofs of the associated lemmas. The proof is done by tracing the cases when the behaviour of machine $\mathsf{M}(S, \mathsf{D})$ is not in accordance with the high-level semantics and checking that the probability of occurrence of such cases is negligible. A more detailed sketch is the following:

- We start by defining variants of $\mathsf{M}(S, \mathsf{D})$ called the defensive variants $\overline{\mathsf{M}}(S, \mathsf{D})$ (Definition C.1). These machines behave like $\mathsf{M}(S, \mathsf{D})$ but include an extra wire where a failure signal is sent whenever the low-level interaction is not in accordance with the high-level semantics. The reader should be aware that these machines are just used as a proof technique, hence there is no need to implement it. All our results are stated in terms of $\mathsf{M}(S, \mathsf{D})$.

- The second step is to create a machine $\widetilde{\overline{\mathsf{N}}^0}(S, \mathsf{D})$ that behaves like $\overline{\mathsf{M}}(S, \mathsf{D})$ but has a common state for all machines $\mathsf{M}_a(S, \mathsf{D})$ (Definition C.5). This is the same as having one single machine that includes all the $\mathsf{M}_a(S, \mathsf{D})$ machines, for all $a \in \mathcal{H}$. We show that $\overline{\mathsf{M}}(S, \mathsf{D})$ is equivalent to $\widetilde{\overline{\mathsf{N}}^0}(S, \mathsf{D})$.

- The third step is to define $\overline{\mathsf{N}}(S, \mathsf{D})$. This is the extreme version of $\overline{\mathsf{N}}^{\widetilde{n}}(S, \mathsf{D})$ where all the encrypted messages are 0's and no signing is ever performed.

Then we have two different arguments. The first is the partial completeness of $\overline{\mathsf{M}}(S, \mathsf{D})$, $\overline{\mathsf{N}}(S, \mathsf{D})$, and the failure of $\overline{\mathsf{N}}(S, \mathsf{D})$.

1. We show that all runs of $\overline{\mathsf{M}}(S, \mathsf{D})$ and $\overline{\mathsf{N}}(S, \mathsf{D})$, where the failure signal is not sent are in conformance with the high-level semantics (Lemma C.6 and Lemma C.7).

2. We show that the probability that the failure signal is issued by $\overline{\mathsf{N}}(S, \mathsf{D})$ machine is negligible by reducing it to the security of the encryption, authentication and signing schemes (Lemma C.8).

The second argument is that $\overline{\mathsf{M}}(S, \mathsf{D})$ is indistinguishable from $\overline{\mathsf{N}}(S, \mathsf{D})$, hence the failure of the former implies the failure of the latter, which only happens with negligible probability. This is done as follows:

1. $\overline{\mathsf{N}}^{\widetilde{n}}(S, \mathsf{D})$ machines are parameterised by $\widetilde{n} = (n_a)_{a \in \mathcal{H}}$. This parameter defines how many messages to each honest principal will be “fake” (a fake message is one where we encrypt 0's instead of the real bitstring). Whenever n_a is reached, it starts behaving like $\overline{\mathsf{M}}_a(S, \mathsf{D})$. For the fake messages we keep an internal table that associates the fake bitstring to the real message so that we can proceed with the correct value when the fake message is provided back to the machine. With a standard cryptographic argument we show that distinguishing $\overline{\mathsf{N}}^{\widetilde{n}}(S, \mathsf{D})$ from $\overline{\mathsf{N}}^{\widetilde{n}+1}(S, \mathsf{D})$, where $\widetilde{n}+1$ has all the components equal to $\widetilde{n}$ except for n_a that we replace by $n_a + 1$ for some principal a (Lemma C.12).

2. We show via a cryptographic argument that for all PPT adversaries, $\overline{\mathsf{M}}(S, \mathsf{D})$ is indistinguishable from $\overline{\mathsf{N}}(S, \mathsf{D})$ (Lemma C.13, C.14, and C.15).

This concludes our proof. □

Finally, our main result states the soundness of equivalence: to show that the machines that implement two stable systems are indistinguishable, it suffices to show that they are safe and bisimilar. We just need an extra condition that the padding size is the same in both cases.

Theorem 3.5 (Soundness for Equivalences). *Let S_1 and S_2 be safe stable systems,* D *a valid shadow for both S_1 and S_2.*

If $S_1 \simeq S_2$, then $\mathsf{M}(S_1, \mathsf{D}) \approx \mathsf{M}(S_2, \mathsf{D})$.

Proof Sketch. For this theorem we also refer the reader to Appendix C for the full proofs of the associated lemmas. The proof is done reusing some of the previous lemmas, in particular Lemma C.15 and with the special Lemmas C.16 and C.17. This lemmas state that for equivalent systems S_1 and S_2 the probabilities of failure of $\overline{\mathsf{N}}(S_1, \mathsf{D})$ and $\overline{\mathsf{N}}(S_2, \mathsf{D})$ are the same up to negligible probability. □

3.7 Related Work

Within formal cryptography, process calculi are widely used to model security protocols. For example, the spi calculus of Abadi and Gordon [AG99] neatly models secret keys and fresh nonces using names and their dynamic scopes. Representing active attackers as pi calculus contexts, one can state (and prove) trace properties and observational equivalences that precisely capture the security goals for these protocols. Automated provers (e.g. [Bla]) also help verify these goals.

In both of these cases, cryptography is supposed to be perfect, that is, decryption is only possible if one knows the encryption key, and an adversary is an arbitrary context that runs in parallel with the specified process. These frameworks provide a formal treatment for the so called Dolev-Yao model.

Abadi, Fournet, and Gonthier develop distributed implementations for variants of the join calculus, with high-level security but no cryptography, roughly comparable to our high-level language. Their implementation is coded within a lower-level calculus with formal cryptography. They establish full abstraction for observational equivalence [Aba98, AFG02, AFG00]. Our approach is similar, but our implementation is considerably more concrete. Also, due to the larger distance between high-level processes and low-level machines, our results are more demanding. Abadi and Fournet also propose a labelled semantics for traffic analysis, in the context of a pi calculus model of a fixed protocol for private authentication [AF04].

Another different approach is to supplement process calculi with concrete probabilistic or polynomial-time semantics. Unavoidably, reasoning on processes becomes more difficult. This was done by Lincoln, Mitchell, Mitchell, and Scedrov [LMMS98]. They introduce a probabilistic process algebra for analysing security protocols, such that parallel contexts coincide with probabilistic polynomial-time adversaries. This was later extended by Mitchell, Ramanathan, Scedrov, and Teague [MRST01, MRST04, MRST06], and Mateus, Mitchell and Scedrov [MMS03]. In the former they develop an equational theory and bisimulation-based proof techniques, while in the latter a general simulatability theorem is presented.

3.8 Conclusions and Future Work

We designed a simple, abstract language for secure distributed communications with two forms of authentication (but no explicit cryptography). Our language provides uniform protection for all messages; it is expressive enough to program a large class of protocols; it also enables simple reasoning about security properties in the presence of active attackers, using labelled traces and equivalences.

We implemented this calculus as a collection of concrete PPT machines embedding standard cryptographic algorithms, and established that low-level PPT adversaries that control their scheduling and the network have essentially the same power as (much simpler) high-level environments. To the best of our knowledge, these are the first cryptographic soundness and completeness results for a distributed process calculus.

We also identified and discussed difficulties that stem from the discrepancy between the two models. Our proofs involve a novel combination of techniques from process calculi and cryptography, but they are less modular than we expected. It would be interesting (and hard) to extend the expressiveness of our calculus, for instance with secrecy and probabilistic choices.

We refer the reader to [AF06b] for the discussion related to soundness and completeness of trace equivalence. There we show soundness of the high-level operational semantics, that is, every series of transitions can be executed (and checked) by a low-level attacker. Said otherwise, the high-level semantics does not give too much power to the environment. As we can characterise any trace using an adversary, we also obtain completeness for trace equivalence: low-level equivalence implies high-level trace equivalence. This result is a corollary of the previous result.

Also, it would be interesting to see if the techniques developed to prove the soundness and completeness results for our calculus can be also applied to similar results for a full-fledge process calculus with explicit cryptography.

Chapter 4

A Process Algebra for Reasoning About Quantum Security

Security protocols are, in general, composed by several agents running in parallel, where each agent computes information (bounded by polynomial-time on the security parameter) and exchange it with other agents. In the context of quantum processes, the computation is bounded by quantum polynomial-time and the information exchanged is supported by qubits. In this Chapter, the problem of defining quantum security properties is addressed using a quantum polynomial-time process algebra. This approach is highly inspired in [MRST01, MMS03].

In Section 2 the process algebra is introduced together with the logarithmic cost random access machine. Both the syntax and the semantics of the process algebra are clearly established, and the section is concluded by presenting the notion of observational equivalence. Section 3 is devoted to emulation and its composition theorem, and finally, in Section 4 quantum zero-knowledge is defined using process emulation.

4.1 Process Algebra

In the context of security protocols it is common to consider a security parameter $\eta \in \mathbb{N}$. In the case of quantum protocols we will also consider such parameter in order to bound the quantum complexity of the principals and adversaries. From now on, the symbol η is reserved to designate such security parameter. The role of this parameter is twofold: it bounds to a polynomial on η the number of qubits that can be sent through channels, and it bounds all the computation to quantum polynomial time (on η). We now detail these aspects culminating with the presentation of the process algebra language.

4.1.1 Quantum polynomial machines

The computational model we adopted to define quantum polynomial machine is based on the logarithmic cost random access machine [CR73] and it is quite similar to the quantum random access machine in [Kni96]. We consider a hybrid model using both classic and quantum memory.

In order to cope with a countable set of qubits qB we adopt the following Hilbert space $\mathcal{H}$ (isomorphic to $\ell^2(2^{qB})$ and $L^2(2^{qB}, \#)$) to model the quantum state (see [MS04, MS06] for a discussion on why $\mathcal{H}$ is the correct Hilbert space for modelling a countable set of qubits):

- each element is a map $|\psi\rangle : 2^{qB} \to \mathbb{C}$ such that:
  - $\mathrm{supp}(|\psi\rangle) = \{v \in 2^{qB} : |\psi\rangle(v) \neq 0\}$ is countable;
  - $\displaystyle\sum_{v \in 2^{qB}} ||\psi\rangle(v)|^2 = \sum_{v \in \mathrm{supp}(|\psi\rangle)} ||\psi\rangle(v)|^2 < \infty$;
- $|\psi_1\rangle + |\psi_2\rangle = \lambda v.\, |\psi_1\rangle(v) + |\psi_2\rangle(v)$;
- $z|\psi\rangle = \lambda v.\, z|\psi\rangle(v)$;
- $\langle\psi_1|\psi_2\rangle = \displaystyle\sum_{v \in V} \overline{|\psi_1\rangle(v)}\, |\psi_2\rangle(v)$.

The inner product induces the norm $|||\psi\rangle|| = \sqrt{\langle\psi|\psi\rangle}$ and so, the distance $d(|\psi_1\rangle, |\psi_2\rangle) = |||\psi_1\rangle - |\psi_2\rangle||$. Clearly, $\{|v\rangle : v \in 2^{qB}\}$ is an orthonormal basis of $\mathcal{H}$ where $|v\rangle(v) = 1$ and $|v\rangle(v') = 0$ for every $v' \neq v$. This basis is called the computational or logic basis of $\mathcal{H}$.

A configuration of a quantum random access machine (QRAM) is triple $\xi = (\mathbf{m}, |\psi\rangle, s)$ where $\mathbf{m} \in \mathbb{N}^{\mathbb{N}}$, $|\psi\rangle \in \mathcal{H}$ and $s \in \mathbb{N}$. The first component of the triple represents the classical memory of the machine—an infinite sequence of natural numbers, the second component represents the quantum state of the machine, and finally the third component is a counter that indicates how many (qu)bit operations are allowed.

We associate to each QRAM a positive polynomial q for bounding the number of allowed (qu)bit operations to $q(\eta)$. In this way, we force each QRAM to terminate in polynomial-time. Given a finite set of qubits at state $|\varphi\rangle$, the *initial configuration* of the QRAM is the triple $\xi_0(|\varphi\rangle) = (\mathbf{m}_0, |\varphi\rangle \otimes |\vec{0}\rangle, q(\eta))$, where the sequence $\mathbf{m}_0$ is such that $\mathbf{m}_0(k) = 0$ for all $k \in \mathbb{N}$ and $|\vec{0}\rangle$ is the unit vector in $\mathcal{H}$ such that $|\vec{0}\rangle(\emptyset) = 1$ (note that if $\mathcal{Q}$ is a 2^n dimension Hilbert space, then there is a canonical isomorphism between $\mathcal{H}$ and $\mathcal{Q} \otimes \mathcal{H}$, and therefore $|\varphi\rangle \otimes |\vec{0}\rangle \in \mathcal{Q} \otimes \mathcal{H}$ can be seen as a unit vector in $\mathcal{H}$). A QRAM receives as input a finite sequence of qubits, but since it is always possible to encode classical bits in qubits this is not a limitation.

The set of atomic commands $\mathcal{AC}$, and their associated cost is presented in the table below[1].

[1]We denote the number of bits required to represent a natural number n by $|n|$.

Number	Instruction	Computational cost
1	$R_i = n$	$\|n\|$
2	$R_i = R_j$	$\|R_j\|$
3	$R_i = R_j + R_k$	$\|R_j\| + \|R_k\|$
4	$R_i = R_j - R_k$	$\|R_j\| + \|R_k\|$
5	$R_i = R_j R_k$	$\|R_j\| \times \|R_k\|$
6	$R_i = R_j / R_k$	$\|R_j\| \times \|R_k\|$
7	$R_i = R_{R_j}$	$\|R_j\| + \|R_{R_j}\|$
8	$R_{R_i} = R_j$	$\|R_i\| + \|R_j\|$
9	$\texttt{Pauli}_X[b]$	1
10	$\texttt{Pauli}_Y[b]$	1
11	$\texttt{Pauli}_Z[b]$	1
12	$\texttt{Hadamard}[b]$	1
13	$\texttt{phase}[b]$	1
14	$\frac{\pi}{8}[b]$	1
15	$\texttt{c-not}[b_1, b_2]$	1
16	$\texttt{measure}[b] \to R_i$	1

Most of the commands above are self-explanatory, but it is worthwhile to notice that all commands are deterministic with exception of `measure`. Indeed, according to the measurement postulates of quantum mechanics (see for instance [CTDL77]), when a quantum system is measured the outcome is stochastic, and moreover the state evolves accordingly to this outcome. Note that we only consider measurements over the computational basis, nevertheless this is not a limitation since any other qubit measurement can be performed by applying a unitary transformation before measuring the qubit over the computational basis.

The set of QRAM *commands* $\mathcal{C}$ is obtained inductively as follows:

1. $a \in \mathcal{C}$ if $a \in \mathcal{AC}$;

2. $c_1; c_2 \in \mathcal{C}$ if $c_1, c_2 \in \mathcal{C}$;

3. $(\texttt{if}\ \ (R_n > 0)\ \ \texttt{then}\ \ c) \in \mathcal{C}$ if $c \in \mathcal{C}$;

4. $(\texttt{while}\ \ (R_n > 0)\ c) \in \mathcal{C}$ if $c \in \mathcal{C}$.

The *execution* of a QRAM command c is a stochastic function between configurations. Let $\Xi = \mathbb{N}^{\mathbb{N}} \times \mathcal{H} \times \mathbb{N}$ be the set of all configurations, and $\text{Prob}_{\text{fin}}(\Xi)$ be the set of all probability measures over $(\Xi, 2^{\Xi})$ such that only a finite set of configurations have probability different from 0. The execution of a QRAM command c is a map $\text{run}_c : \Xi \longrightarrow \text{Prob}_{\text{fin}}(\Xi)$, and we write $[c]\ \xi \to_p \xi'$ to denote that $\text{Pr}_{\text{run}_c(\xi)}(\xi') = p$. The execution of QRAM commands can be defined using the following rules, which are quite intuitive:

$$\frac{s \geq |n|}{[R_i = n]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - |n|)}\ (R_i = n),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = n$;

$$\frac{s \geq |R_j|}{[R_i = R_j] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - |R_j|)} \; (R_i = R_j),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j)$;

$$\frac{s \geq |R_j| + |R_k|}{[R_i = R_j + R_k] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_k|))} \; (R_i = R_j + R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j) + \mathbf{m}(k)$;

$$\frac{s \geq |R_j| + |R_k|}{[R_i = R_j - R_k] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_k|))} \; (R_i = R_j - R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \max(\mathbf{m}(j) - \mathbf{m}(k), 0)$;

$$\frac{s \geq |R_j| \times |R_k|}{[R_i = R_j R_k] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| \times |R_k|))} \; (R_i = R_j R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j)\mathbf{m}(k)$;

$$\frac{s \geq |R_j| \times |R_k|}{[R_i = R_j / R_k] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| \times |R_k|))} \; (R_i = R_j / R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \lfloor \mathbf{m}(j) / \mathbf{m}(k) \rfloor$;

$$\frac{s \geq |R_j| + |R_{R_j}|}{[R_i = R_{R_j}] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_{R_j}|))} \; (R_i = R_{R_j}),$$

where $\mathbf{m}'(k) = \mathbf{m}'(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(\mathbf{m}(j))$;

$$\frac{s \geq |R_i| + |R_j|}{[R_{R_i} = R_j] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_i| + |R_j|))} \; (R_{R_i} = R_j),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq \mathbf{m}(i)$ and $\mathbf{m}'(\mathbf{m}(i)) = \mathbf{m}(j)$;

$$\frac{s \geq 1}{[\mathtt{Pauli}_X[b]] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi'\rangle, s - 1)} \; (\mathtt{Pauli}_X[b]),$$

where $|\psi'\rangle$ is obtained from $|\psi\rangle$ by applying the Pauli$_X$ operator $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ on qubit b. Similar rules apply to the following one-qubit operators:

$$\text{Pauli}_Y \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \qquad \text{Pauli}_Z \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix};$$

$$\text{Hadamard } \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \qquad \text{Phase } \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \qquad \tfrac{\pi}{8}\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix};$$

$$\frac{s \geq 1}{[\texttt{c-not}[b_1, b_2]] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi'\rangle, s-1)} \ (\texttt{c-not}[b_1, b_2]),$$

where $|\psi'\rangle$ is obtained from $|\psi\rangle$ by applying the control-not operator

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

on qubits b_1 and b_2;

$$\frac{s \geq 1}{[\texttt{measure}[b] \rightarrow R_i] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s-1))} \ (\texttt{measure}[b] \rightarrow R_i = 0),$$

where $|\psi'\rangle$ is equal to $\frac{P_0|\psi\rangle}{|P_0|\psi\rangle|}$, $p = |P_0|\psi\rangle|$ (P_0 is the projector onto the subspace of $\mathcal{H}$ where qubit b takes value $|0\rangle$), $\mathbf{m}'(i) = 0$ and $\mathbf{m}'(j) = \mathbf{m}(j)$ for all $j \neq i$;

$$\frac{s \geq 1}{[\texttt{measure}[b] \rightarrow R_i] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s-1))} \ (\texttt{measure}[b] \rightarrow R_i = 1),$$

where $|\psi'\rangle$ is equal to $\frac{P_1|\psi\rangle}{|P_1|\psi\rangle|}$, $p = |P_1|\psi\rangle|$ (P_1 is the projector onto the subspace of $\mathcal{H}$ where qubit b takes value $|1\rangle$), $\mathbf{m}'(i) = 1$ and $\mathbf{m}'(j) = \mathbf{m}(j)$ for all $j \neq i$;

$$\frac{[c_1] \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_{p_1} (\mathbf{m}', |\psi'\rangle, s') \qquad [c_2] \ (\mathbf{m}', |\psi'\rangle, s') \rightarrow_{p_2} (\mathbf{m}'', |\psi''\rangle, s'')}{[c_1; c_2] \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_{p_1 \times p_2} (\mathbf{m}'', |\psi''\rangle, s'')} \ (c_1; c_2);$$

$$\frac{\mathbf{m}(n) > 0 \qquad s \geq |R_n| \qquad [c] \ \ (\mathbf{m}, |\psi\rangle, s - |R_n|) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s')}{[(\texttt{if} \ \ (R_n > 0) \ \ \texttt{then} \ \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_p [c] \ \ (\mathbf{m}', |\psi'\rangle, s')} \ (\texttt{if}\top);$$

$$\frac{\mathbf{m}(n) = 0}{[(\texttt{if} \ \ (R_n > 0) \ \ \texttt{then} \ \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi\rangle, s)} \ (\texttt{if}\bot);$$

$$\frac{\begin{array}{c} \mathbf{m}(n) > 0 \qquad s \geq |R_n| \\ [c; (\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s - |R_n|) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s') \end{array}}{[(\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s')} \ (\texttt{while}\top);$$

$$\frac{\mathbf{m}(n) = 0}{[(\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi\rangle, s - |R_n|)} \ (\texttt{while}\bot).$$

Observe, that the reduction of QRAM commands always terminate, since every computation is bounded by $q(\eta)$ (qu)bit steps. The execution of a QRAM command can be seen as a word run of a quantum automata [MMS05], however a detailed discussion about this subject is out of the scope of this abstract.

The output of a QRAM is the quantum state of a set of qubits. This output set is determined by another positive polynomial o associated to the machine. Given a security parameter η, the set of output qubits is constituted by the first $o(\eta)$ qubits.

Definition 4.1. A *quantum polynomial machine* is a triple $M = (c, q, o)$ where c is a QRAM command, q is a positive step bounding polynomial and o is a positive output polynomial. We denote the set of all these triples by QPM.

Given a quantum polynomial machine M and a security parameter η, the computation of M over state $|\psi\rangle$ is the probability distribution over the state of the first $o(\eta)$ qubits of $|\psi'\rangle$, where this distribution is defined by the execution rules $[c](\mathbf{m}_0, |\psi\rangle, q(\eta)) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s')$. Hence, the computation of a QRAM is a probability distribution over the state space of the first $o(\eta)$ qubits. It is traditional in quantum algorithms to measure all relevant qubits at the end of the computation in order to obtain a classical result (see Shor's and Grover's algorithms). However, since we use QRAM to compute quantum information that can be sent through quantum channels, we do not impose this final measurement since it may be desirable to send a superposition through a quantum channel.

The following result asserts that the QRAM model is equivalent to the usual quantum circuit computational model (a careful presentation of this result is out of the scope of this abstract).

Proposition 4.1. *For any uniform family of polynomial quantum circuits $Q = \{Q_\eta\}_{\eta\in\mathbb{N}}$, there exists a quantum polynomial machine M_Q such that the M_Q computes the same stochastic function as Q. Moreover, for any quantum polynomial machine M there exists an equivalent uniform family of polynomial quantum circuits $Q_M = \{Q_\eta\}_{\eta\in\mathbb{N}}$.*

Proof. Proof (Sketch): Note that a uniform circuit uses precisely the gates defined as quantum atomic commands of the QRAM. The construction of the circuit can be mimicked by a RAM command c. Since this construction must be polynomial in η, the program must terminate in polynomial time and therefore, there is a polynomial q to bound the number of steps, finally the output must always be a polynomial set of qubits, and therefore we are able to construct an equivalent QRAM machine.

On the other hand a QRAM program is the realisation of the uniform family construction, since, for each η, a circuit can be retrieved by looking at the finite (do not forget that QRAM programs always terminate) sequence of quantum atomic gates generated by the execution of the command. The stochastic nature of the execution does not bring a problem, since gates placed after a measurement can be controlled by the outcome of that measurement. If a measurement gives the value 1 to a qubit and in that case a gate U is placed at some qubit b, then the circuit should be constructed by placing a control-U gate controlled by the measured qubit and targeted at b. $\square$

4.1.2 Process algebra

As stated before, we require to know who possesses a qubit in order to know who can retrieve some piece of information. In order to deal with this fact, a qubit is considered to belong to some agent, and therefore, the set of qubits qB is partitioned among all agents. To make this more precise, a countable set $A = \{a_1, \ldots, a_k, \ldots\}$ of agents is fixed once and for all, and moreover the partition $q\mathcal{B} = \{qB_{a_i}\}_{a_i \in A}$ of qB is such that each set qB_{a_i} is countable and recursively enumerable.

Note that each qB_{a_i} has a total order (with a bottom element) induced by its recursive enumeration. The purpose of this total ordering is to reindex the qubits accessed by a QPM M when an agent a executes M. An obvious desideratum of the system is that an agent a is restricted to compute over its own qubits qB_a, and therefore, when agent a executes a quantum polynomial machine M, this machine must have access only to the qubits in qB_a (note that if the qubits of a are entangled with other qubits, then when the former are modified so can be the latter). Therefore, if, for instance, an agent a executes a machine that consists of the command $\texttt{Pauli}_X[b]$, and if qB_a is recursively enumerated by γ, then the command effectively executed is $\texttt{Pauli}_X[\gamma(b)]$. The same procedure applies to the input and output qubits, so when a machine executed by a outputs the first $o(\eta)$ qubits, the machine is in fact outputting the qubits $\{\gamma(o(1)), \ldots, \gamma(o(\eta))\} \subset qB_a \subseteq qB$.

Communication between agents is achieved via public channels, allowing qubits to be exchanged. Clearly, this process is modelled by modifying the partition of qB. It is also convenient to allow parallelism inside an agent (that is, an agent may be constituted by several processes in parallel), for this purpose, private channels (that cannot be intercepted) allowing communication between the agent local processes are introduced. To make this assumptions clear, two countable disjoint sets of *quantum channels* are considered, the set of *global or public channels* $G = \{g_1, g_2, \ldots, g_k, \ldots\}$, and the set of *local or private channels* $L = \{l_1, l_2, \ldots, l_k, \ldots\}$. We denote by C the set $G \cup L$. All global channels can be read and written by an adversary while local channels correspond to private communication from one agent to itself. One role of the security parameter is to bound the bandwidth of the channels. Hence, we introduce a *bandwidth map* $\mathbf{bw} : C \to \mathbf{q}$, where $\mathbf{q}$ is the set of all polynomials taking positive values. Given a value η for the security parameter, a channel c can send at most $\mathbf{bw}(c)(\eta)$ qubits.

We also consider a countable set of variables *Var* $= \{x_1, x_2, \ldots, x_k, \ldots\}$, which are required to define qubit terms. A qubit term t is either a finite subset of qB or a variable $x \in$ *Var*.

Finally, we present the language of processes, which is a fragment of π-calculus. Mind that the overall computation must be quantum polynomial on η and therefore we do not cope with recursion nor mobility. First, we establish the language of an agent, that we call local process language.

Definition 4.2. The *language of local processes* $\mathcal{L}$ is obtained inductively as follows:

1. $0 \in \mathcal{L}$ (termination);

2. $c\langle M(t)\rangle \in \mathcal{L}$ where $M \in$ QPM, t is a qubit term, and $c \in C$ (output);

3. $c(x).Q \in \mathcal{L}$ where $c \in C$, $x \in$ *Var* and $Q \in \mathcal{L}$ (input);

4. $[M(t) = 0].Q$ where $M \in$ QPM, t is a qubit term, and $Q \in \mathcal{L}$ (match);

5. $(Q_1|Q_2)$ where $Q_1, Q_2 \in \mathcal{L}$ (parallel composition);

6. $!_q Q$ where $Q \in \mathcal{L}$ and $q \in \mathbf{q}$ (bounded replication).

Most of the (local) process terms are intuitive. The output term $c\langle M(qB')\rangle$ means that the output of machine M, which received the finite set of qubits qB' as input, is sent through channel c. The input term $c(x).Q$ means that a set of qubits is going to be received on c, and upon reception, x takes the value of the received qubits.

After fixing the security parameter η, we can get rid of replication by evaluating each process $!_q R$ as $q(\eta)$ copies of R in parallel. Therefore, we always assume that a process term has no replication. Now, as state before, a protocol is constituted by a set o agents running in parallel, therefore the global language (or protocol language) is quite simple:

Definition 4.3. The *language of global processes* $\mathcal{G}$ over a set of agents A is defined inductively as follows:

1. $\mathbf{0} \in \mathcal{G}$ (global termination);

2. $P \| (a : Q) \in \mathcal{G}$ where $P \in \mathcal{G}$, $a \in A$ does not occur in P, and $Q \in \mathcal{L}$ (global parallel composition).

The following example uses the process language to describe the RSA cryptanalysis using Shor's algorithm.

Example 4.1 (Shor's based RSA cryptanalysis). Let p, q be primes (with η length binary expansion), and e, d integers such that $ed \equiv 1 \mod \phi(pq)$. Alice is a simple process A that knows some message w and outputs $w^e \mod pq$, where e is the public key of Bob. This dummy process can be presented as

$$(a : A(w)) := (a : g\langle w^e \mod pq\rangle).$$

Bob receives x and computes $x^d \mod pq$. This procedure can be modelled by the following process:

$$(b : B) := (b : g(x).(l\langle x^d \mod pq\rangle | l(y).0)).$$

Therefore the RSA protocol is given by the process $(a : A(w)) \| (b : B)$. Finally, we can write the "attacking" process, Eve. She factorises pq, inverts $e \mod \phi(pq)$ (thus, allowing her to find d), and intercepts the message sent by Alice (on channel g). We write this process as follows:

$$(c : l_1\langle \text{Shor}(pq)\rangle | l_1(y).l_2\langle \text{Inv}(y, e)\rangle | g(x).l_2(z).(l_3\langle x^z \mod pq\rangle | l_3(w).0)).$$

4.1.3 Semantics

In order to define the semantics of a local process we need to introduce the notion of local configuration. A *local configuration* or *agent configuration* is a triple $(|\psi\rangle, qB_a, Q)$ where $|\psi\rangle \in \mathcal{H}$, $qB_a \subseteq qB$ is a countable, recursive enumerable set and $Q \in \mathcal{L}$. The first element of the local configuration is the global state of the protocol, the second element is the set of qubits the agent possesses and the last element is the local process term.

The semantics of a local process is a probabilistic transition system where the transitions are defined by rules. We use $(|\psi\rangle, qB_a, Q) \rightarrow_p (|\psi'\rangle, qB_a, Q')$ to state that, at global state $|\psi\rangle$, when agent a possesses qubits qB_a, the local process Q is reduced to Q' and global state is modified to $|\psi'\rangle$ with probability p. It is also worthwhile to observe that we use the notation $M(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2)$ to denote that the execution of the QRM M, operating on qB_a (that is, using the recursive enumeration of qB_a to reindex the position of the qubits), and receiving as input qB_1, outputs qB_2 and modifies the global state $|\psi\rangle$ to $|\psi'\rangle$ with probability p. For the case of local processes, the sets qB_1 and qB_2 are irrelevant, because the qubits owned by the agent remain the same when a local communication (LCom rule) is applied. Their functionality will be clear when we present the global rules.

$$\frac{M(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad qB_1, qB_2 \subseteq qB_a \quad |qB_2| \leq \mathbf{bw}(l)(\eta)}{(|\psi\rangle, qB_a, l(x).Q|l\langle M(qB_1)\rangle) \rightarrow_p (|\psi'\rangle, qB_a, Q^x_{qB_2})} \text{ (LCom)}$$

We also introduce the term $M;\text{Meas}$ to denote the machine that, after executing M performs a measurement on the computational basis of the output qubits of M. So a match corresponds to performing a measurement on the output qubits of M and checking whether the result is the 0 word.

$$\frac{(M;\text{Meas})(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad |\psi'\rangle|_{qB_2} = |\vec{0}\rangle}{(|\psi\rangle, qB_a, [M(qB_1) = 0].Q) \rightarrow_p (|\psi'\rangle, qB_a, Q)} \text{ (Match}\top\text{)}$$

$$\frac{(M;\text{Meas})(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad |\psi'\rangle|_{qB_2} \neq |\vec{0}\rangle}{(|\psi\rangle, qB_a, [M(qB_1) = 0].Q) \rightarrow_p (|\psi'\rangle, qB_a, \mathbf{0})} \text{ (Match}\bot\text{)}$$

The remaining rules are self-explanatory.

$$\frac{(|\psi\rangle, qB_a, P) \rightarrow_p (|\psi'\rangle, qB_a, P')}{(|\psi\rangle, qB_a, P|Q) \rightarrow_p (|\psi'\rangle, qB_a, P'|Q)} \text{ (LLPar)}$$

$$\frac{(|\psi\rangle, qB_a, Q) \rightarrow_p (|\psi'\rangle, qB_a, Q')}{(|\psi\rangle, qB_a, P|Q) \rightarrow_p (|\psi'\rangle, qB_a, P|Q')} \text{ (LRPar)}$$

We proceed by presenting the global rules. A *global configuration* is a triple $(|\psi\rangle, q\mathcal{B}, P)$ where $|\psi\rangle \in \mathcal{H}$, $q\mathcal{B} = \{qB_a\}_{a\in A}$ is a partition of qB indexed by the set of agents A (where each qB_a is countable and r.e.) and $P \in \mathcal{G}$. The semantics of a global process is defined by the following rules:

$$\frac{(|\psi\rangle, qB_a, Q) \rightarrow_p (|\psi'\rangle, qB_a, Q')}{(|\psi\rangle, q\mathcal{B}, (a : Q)) \rightarrow_p (|\psi'\rangle, q\mathcal{B}, (a : Q))} \text{ (LtoG)}$$

$$\frac{M(|\psi\rangle, qB_b, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad qB_1, qB_2 \subseteq qB_b \quad |qB_2| \leq \mathbf{bw}(g)(\eta)}{(|\psi\rangle, q\mathcal{B}, (a : g(x).Q)\|(b : g\langle M(qB_1)\rangle)) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', (a : Q^x_{qB_2}))} \text{ (GCom)}$$

where $q\mathcal{B}' = \{qB'_a\}_{a\in A}$, $qB'_a = qB_a \cup qB_2$, $qB'_b = qB_b \setminus qB_2$, and $qB'_c = qB_c$ for all $c \neq a, b$.

$$\frac{(|\psi\rangle, q\mathcal{B}, P_1) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', P'_1)}{(|\psi\rangle, q\mathcal{B}, P_1\|P_2) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', P'_1\|P_2)} \text{ (GLPar)}$$

$$\frac{(|\psi\rangle, q\mathcal{B}, P_2) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', P'_2)}{(|\psi\rangle, q\mathcal{B}, P_1\|P_2) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', P_1\|P'_2)} \text{ (GRPar).}$$

All the rules are very simple to grasp. The only non trivial rule is global communication (GCom), that makes qubits to be exchanged from one agent to another, and therefore an adjustment is required in the qubit partition.

Process term reductions are non-deterministic, in the sense that several different reductions could be chosen at some step. In order to be possible to make a quantitative analysis, this reduction should be probabilistic. For the sake of simplicity, we assume a uniform scheduler, that is, the choice on any possible reduction is done with uniform probability over all possible non-deterministic reductions. We do not present in detail the scheduler model but, in principle, any probability distribution modelled by a QPM can be used to model the scheduler policy. Finally, note that by applying local and global rules, and assuming a uniform scheduler, one can define the many step reduction $\rightarrow^*_p$ such that $(|\psi_1\rangle, q\mathcal{B}_1, P_1) \rightarrow^*_p (|\psi_n\rangle, q\mathcal{B}_n, P_n)$, whenever:

- $(|\psi_1\rangle, q\mathcal{B}_1, P_1) \rightarrow_{p_1} (|\psi_2\rangle, q\mathcal{B}_2, P_2) \rightarrow_{p_2} \cdots \rightarrow_{p_{n-1}} (|\psi_n\rangle, q\mathcal{B}_n, P_n)$;
- $p = \frac{p_1}{R_1} \times \frac{p_2}{R_2} \times \cdots \times \frac{p_{n-1}}{R_{n-1}}$ where R_i is the number of possible non-deterministic choices for $(|\psi_i\rangle, q\mathcal{B}_i, P_i)$ for all $i \in \{1, \ldots, n-1\}$;
- $(|\psi_n\rangle, q\mathcal{B}_n, P_n)$ cannot be reduced any more.

The many step reduction takes into account the scheduler choice, by weighting each stochastic reduction p_i with yet another probability $\frac{1}{R_i}$, where R_i is the number of possible non-deterministic choices at step i.

4.1.4 Observations and observational equivalence

At the end of a protocol, each agent $a \in A$ is allowed to measure a polynomial (in η) number of qubits in qB_a to extract information. We can always assume that these qubits are the first, say, $r(\eta)$ qubits of qB_a where r is a positive polynomial. Therefore, the many step reduction of a process term P induces a probability distribution on $2^{r(\eta)}$, where $2^{r(\eta)}$ is the set of all possible outcomes of $r(\eta)$ qubits when measured over the computational basis (that is, $2^{r(\eta)}$ is the set of all $r(\eta)$-long binary words).

Definition 4.4. Given a positive polynomial r and a global configuration $(|\psi\rangle, q\mathcal{B}, P)$, let

$$\Gamma_{(|\psi\rangle, q\mathcal{B}, P)} = \{(|\psi'\rangle, q\mathcal{B}', P') : (|\psi\rangle, q\mathcal{B}, P) \rightarrow^*_p (|\psi'\rangle, q\mathcal{B}', P') \text{ and } p > 0\}.$$

We define the *observation of an agent* a to be the family of probability measures

$$O^a_r = \{(2^{r(\eta)}, 2^{2^{r(\eta)}}, \mathrm{Pr}^a_{r(\eta)})\}_{\eta \in \mathbb{N}}$$

where:

- $\mathrm{Pr}^a_{r(\eta)}(\{w\}) = \sum_{\gamma \in \Gamma_{(|\psi\rangle, q\mathcal{B}, P)}} p_\gamma \times |\langle w|\psi_\gamma\rangle|$;
- p_γ is such that $(|\psi\rangle, q\mathcal{B}, P) \rightarrow^*_{p_\gamma} \gamma$;
- $|\psi_\gamma\rangle$ is the first component of γ;
- $|\langle w|\psi_\gamma\rangle|$ is the probability of observing the $r(\eta)$-long binary word w by measuring the $r(\eta)$ first qubits of qB_a (qubits in possession of agent a) of $|\psi_\gamma\rangle$ in the computational basis.

Note that the summation used to compute $\mathrm{Pr}^a_{r(\eta)}(\{w\})$ is well defined, since $\Gamma_{(|\psi\rangle, q\mathcal{B}, P)}$ is finite. It is clear at this point,that an observation of an agent is a random $r(\eta)$-long binary word, with distribution given by $\mathrm{Pr}^a_{r(\eta)}$.

The notion of observational equivalence we adopt is based on computational indistinguishability as usual in the security community [MRST01]. First, we introduce the concept of context. The set of *global contexts* $\mathcal{C}$ is defined inductively as follows: $[\ ] \in \mathcal{C}$; $C[\ ]\|P$ and $P\|C[\ ] \in \mathcal{C}$ provided that $C[\ ] \in \mathcal{C}$ and $P \in \mathcal{G}$. Given a context $C[\ ]$ and a global process P, the notation $C[P]$ means that we substitute the process P for the $[\ ]$ in $C[\ ]$.

Definition 4.5. Let P and P' be process terms. We say that P is *computationally indistinguishable by agent* a from P' if and only if for every context $C[\ ]$, polynomials q and r, $|\psi\rangle \in \mathcal{H}$, partition $q\mathcal{B}$ of qB, η sufficiently large and binary word $w \in 2^{r(\eta)}$,

$$|\mathrm{Pr}^a_{r(\eta)}(w) - \mathrm{Pr}'^a_{r(\eta)}(w)| \leq \frac{1}{q(\eta)}$$

where $\mathrm{Pr}^a_{r(\eta)}$ is given by the observation of a for configuration $(|\psi\rangle, q\mathcal{B}, C[P])$ and $\mathrm{Pr}'^a_{r(\eta)}$ is given by the observation of a for configuration $(|\psi\rangle, q\mathcal{B}, C[P'])$. In such case we write $P \simeq P'$.

Two processes are computationally indistinguishable if they are indistinguishable by contexts, that is, for any input (here modelled by $|\psi\rangle$ and $q\mathcal{B}$), there is no context which can distinguish, up to a negligible function, the outputs produced. The definition above extends the classical definition of computational indistinguishability to the quantum case, since processes can be modelled by quantum polynomial machines and therefore $C[\ ]$ induces the required distinguishing machine. A detailed proof of this result is out of the scope of this extended abstract.

In order to set up compositionality, the following result is of the utmost importance:

Proposition 4.2. *Computational indistinguishability is a congruence relation with respect to the parallel primitive of* $\mathcal{G}$.

Proof. Both symmetry and reflexivity are trivial to check. Transitivity follows by triangular inequality, and taking into account that $\frac{1}{2}q(n)$ is a polynomial. Congruence on the global parallel operator follows by noticing that for any contexts $C[\ ]$ and $C'[\ ]$, $C'[C[\ ]]$ is also a context. $\square$

4.2 Emulation and Composition Theorem

One of the most successful ways for defining secure concurrent cryptographic tasks is via process emulation [AG99, Can00]. This definitional job boils down to the following: a process realises a cryptographic task if and only if it emulates an ideal process that is known to realise such task. In this section, guided by the goal of defining secure functionalities, we detail the notion of emulation for the quantum process calculus defined in the previous section.

Let I be an ideal protocol that realises (the honest part of) some secure protocol and P a process that implements the functionality specified by I. The overall goal is to show that P realises, without flaws, (part of) the secure functionality specified by I. The goal is achieved if for any real adversary, say $(a : A)$, the process $P||(a : A)$ is computationally indistinguishable by the adversary a from the process $I||(a : B)$ for some ideal adversary $(a : B)$, where an ideal adversary is an adversary which cannot corrupt I and a real adversary is any local process for agent a. This property asserts that given a real adversary $(a : A)$, agent a cannot distinguish the information leaked by $P||(a : A)$ from the information leaked by the well behaved process $I||(a : B)$ for some ideal adversary $(a : B)$, and therefore, we infer that $P||(a : A)$ is also well behaved. This discussion leads to the concept of emulation with respect to a set of real adversaries $\mathcal{A}$ and ideal adversaries $\mathcal{B}$.

Definition 4.6. Let P and I be process terms and $\mathcal{A}$ and $\mathcal{B}$ sets of global processes where the only agent is the adversary a, then P *emulates* I with respect to $\mathcal{A}$ and $\mathcal{B}$ if and only if for all processes $(a : A) \in \mathcal{A}$ there exists a process $(a : B) \in \mathcal{B}$ such that $P||(a : A) \simeq I||(a : B)$. In such case we write $P \equiv^a_{\mathcal{A},\mathcal{B}} I$ and say that P is a secure implementation of I with respect to $\mathcal{A}$ and $\mathcal{B}$.

A desirable property of the emulation relation is the so called Composition Theorem. This result was first discussed informally for the classical secure computation setting in [MR91], and states the following: if P is a secure implementation of part I of an ideal protocol, R and J are two protocols which use the ideal protocol I as a component, and finally, R is a secure implementation of J, then R^I_P should be a secure implementation of J. This result is captured as follows:

Theorem 4.3. *Let P, I be processes, $R[\ ]$ and $J[\ ]$ contexts and $\mathcal{A}, \mathcal{B}$ sets of processes over agent a and $\mathcal{C}, \mathcal{D}$ sets of processes over agent b. If $R[I||(a : B)] \equiv^b_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$ for any $(a : B) \in \mathcal{B}$ and $P \equiv^a_{\mathcal{A},\mathcal{B}} I$ then for any adversary $(a : A) \in \mathcal{A}$ there exists $(a : B) \in \mathcal{B}$ such that $R[Q||(a : A)] \equiv^b_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$.*

Proof. Let $(a : A) \in \mathcal{A}$ and $(a : B) \in \mathcal{B}$ be such that $P||(a : A) \simeq I||(a : B)$. Now choose some $(b : C) \in \mathcal{C}$, clearly, $R[Q||(a : A)]||(c : C) \simeq R[I||(a : B)]||(c : C)$ since $\simeq$ is a congruence relation. Moreover, since $R[I||(a : B)] \equiv_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$, there is a $(b : D) \in \mathcal{D}$ such that $R[I|(a : B)]||C \simeq J[I||(a : B)]||(b : D)$. Finally, by transitivity of $\simeq$, we have that $R[Q||(a : A)]||(b : C) \simeq J[I||(a : B)]||(b : D)$ and hence $R[Q||(a : A)] \equiv_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$. $\square$

Observe that ideal protocols are constituted by a honest part I and an ideal adversary $(a : B)$, and therefore are of the form $I||(a : B)$. This justifies why $R[I||(a : B)]$ was considered in the proposition above instead of $R[I]$. Moreover, adversaries for the functionality implemented by R and J might be different from those of I and Q, therefore, two pairs of sets of processes $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{A}$, $\mathcal{B}$ are required to model two kinds of adversaries.

4.3 Quantum Zero-Knowledge Proofs

An interactive proof is a two party protocol, where one agent is called the *prover* and the other is called the *verifier*. The main objective of the protocol is to let the prover convince the verifier of the validity of an assertion, however, this must be done in such a way that the prover cannot convince the verifier of the validity of some false assertion.

Any interactive proof system fulfills two properties: completeness and soundness. Completeness states that if the assertion the prover wants to convince the verifier is true, then the verifier should be convinced with probability one. On the other hand, soundness is fulfilled if the verifier cannot be convinced, up to a negligible probability, of a false assertion. Therefore, completeness and soundness allow the verifier to check whether the assertion of the prover is true or false.

Zero-knowledge is a property of the prover (strategy). Consider the following informal notion of (quantum) computational zero-knowledge strategy, which corresponds to the straightforward lifting to the quantum setting of the classical version:

Definition 4.7. A prover strategy S is said to be *quantum computational zero-knowledge* over a set L if and only if for every quantum polynomial-time verifier strategy, V there exists quantum polynomial-time algorithm M such that $(S, V)(l)$ is (quantum) computationally indistinguishable from $M(l)$ for all $l \in L$, where (S, V) denotes the output of the interaction between S and V.

The main application of zero-knowledge proof protocols in the cryptographic setting is in the context of a user U that has a secret and is supposed to perform some steps, depending on the secret. The problem is how can other users assure that U has carried out the correct steps without U disclosing its secret. Zero-knowledge proof protocols (ZKP) can be used to satisfy these conflicting requirements.

Zero-knowledge essentially embodies that the verifier cannot gain more knowledge when interacting with the prover than by running alone a quantum polynomial time program (using the same input in both cases). That is, running a the verifier in parallel with the prover should be indistinguishable of some quantum polynomial time program.

Actually, the notion of (quantum computational) zero-knowledge proofs can be captured through emulation very easily. Assuming that a proof strategy $S(x)$ and verifier $V(x)$ are modelled as terms of the process algebra, it is actually possible to model the interaction between p and v by the process $(p : S)||(v : V)$. Denote by $\mathcal{L}^v(l)$ the set of all process terms for the verifier $(v : V)^x_l$, that is, any process term $(v : V)$ where the free variable x was replaced by the binary word l. We have the following characterisation:

Proposition 4.4. *A process term $(p : S)$ denoting a proof strategy is computational zero-knowledge for L if and only if $(p : S)^x_l \equiv^v_{\mathcal{L}^v(l),\mathcal{L}^v(l)} 0$, for all $l \in L$.*

Proof. Proof (Sketch): Notice that the ZKP resumes to impose that for all $(v : V)_l^x$ there is a process $(v : V')_l^x$ such that $(p : S)_l^x||(v : V)_l^x \simeq 0||(v : V')_l^x$. Since the semantics of a local process can be modelled by a QPM, and moreover $0||(v : V')_l^x$ can model any QPM, the characterisation proposed in this proposition is equivalent to Definition 4.7. □

So, a process $(p : S)$ models a quantum zero-knowledge strategy if, from the point of view of the verifier, it is impossible to distinguish the final result of the interaction with $(p : S)$ from the interaction with the 0 process. A clear corollary of Theorem 4.3 is that, quantum zero-knowledge is compositional.

It is simple to adapt the emulation approach to several other quantum security properties, like quantum secure computation, authentication and so on.

4.4 Conclusions and Future Work

The contributions of this work are multiple. First, we introduced a process algebra for specifying and reasoning about (quantum) security protocols. To restrict the computation power of the agents to quantum polynomial-time, we introduced the logarithm cost quantum random access machine, and incorporated it in the process language. Due to the special aspects of quantum information, qubits were assumed to be partitioned among agents, and the (quantum) computation of an agent was restricted to its own qubits.

Second, we defined observational equivalence and quantum computational indistinguishability for the process algebra at hand, and showed that the latter is a congruence relation. Moreover, we obtained a simple corollary of this result: security properties defined via emulation are compositional.

Finally, we illustrated the definition of a security property via emulation with the concept of quantum zero-knowledge. It is however straightforward to adapt this approach to several other quantum security properties, like quantum secure computation.

As for future work, a research program can be set up by just extending several results of classical process algebras to the quantum world. For instance, one can assume that adversaries have quantum polynomial time power and try to find a sound and complete implementation of a process algebra over such quantum computational model.

Chapter 5

Conclusions and Future Work

To conclude this dissertation we are going to summarise its main contributions and point out some of the directions in which this work may be extended.

5.1 Summary of Contributions

This dissertation is divided in three main chapters. In Chapter 2 we discussed extensions of the Abadi-Rogaway logics of indistinguishability [AR02]. In more detail, we dealt with two restrictions of their results, existence of key-cycles in the encrypted expressions, and partial leakage of information.

Soundness in the presence of key-cycles was never dealt in previous results and seemed to be a gap between the symbolic and the computational models. In fact, our results show that in order to bridge this gap we need to perform changes in the computational definitions, in contrast with the usual practice of changing the symbolic model. We use the notion of KDM-security [BRS02] and show that, when the encryption scheme is KDM-secure, it is possible to obtain soundness even in the presence of key-cycles. We also show that the computational soundness property neither implies nor is implied by type-0 security, and thus the original Abadi-Rogaway result could not have been demonstrated for key-cycles using the security notions described in their work. In Appendix B we show similar results for public-key encryption and in particular that computational soundness property neither implies nor is it implied by security against chosen ciphertext attack, CCA-2.

The other weakness of the Abadi-Rogaway results that we discuss in this dissertation is the case of partial leakage of information by an encryption scheme. The original result assumed a very strong notion of security (type-0) which is not actually achieved by many encryption schemes. Thus, one might wonder if a similar result might be derived for weaker schemes. We have showed that for symmetric encryption, subtle differences between security definitions can be faithfully reflected in the formal symbolic setting. To this end, we introduced a general probabilistic framework which includes both the computational and the information-theoretic encryption schemes as special cases. We have established soundness and completeness theorems of formal encryption in this general framework, as well as new applications to specific settings:

an information-theoretic interpretation of formal expressions in One-Time Pad, and also computational interpretations in type-1 (length-revealing), type-2 (which-key revealing) and type-3 (which-key and length revealing) encryption schemes based on computational complexity.

In Chapter 3 we presented an approach for the study of sound abstractions of cryptography using process algebras. As process algebras have been widely used in the study of security of concurrent systems, all the existing results are stated in terms of the Dolev-Yao Model, hence no real cryptographic guarantees are achieved. In more detail, we designed a simple, abstract language for secure distributed communications with two forms of authentication (but no explicit cryptography). Our language is expressive enough to program a large class of protocols, and enables simple reasoning about security properties in the presence of active attackers, using labelled traces and equivalences. We provide a concrete implementation for this calculus as a collection of concrete PPT machines embedding standard cryptographic algorithms, and established that low-level PPT adversaries that control their scheduling and the network have essentially the same power as (much simpler) high-level environments. To the best of our knowledge, this is the first cryptographic soundness and completeness results for a distributed process calculus.

In Chapter 4 we introduced a process algebra for specifying and reasoning about (quantum) security protocols. To the best of our knowledge, this was the first quantum programming language that was fully designed having in mind the specific subtleties of security protocols. In order to restrict the computation power of the agents to quantum polynomial-time, we introduced the logarithm cost quantum random access machine, and incorporated it in the process language. We also defined observational equivalence and quantum computational indistinguishability for the process algebra at hand, and showed that the latter is a congruence relation. We obtained a simple corollary that security properties defined via emulation are compositional. As an application we illustrated the definition of a security property via emulation with the concept of quantum zero-knowledge. It is however straightforward to adapt this approach to several other quantum security properties, like quantum secure computation.

5.2 Limitations of Our Results and Future Work

A limitation of our results of Chapter 2 regarding key-cycles is that the only known realisation of a KDM-encryption scheme is in the Random-Oracle Model. In spite of our results do not depend upon this realisation (our results only use the KDM-scheme as a black-box scheme) we would like to design an KDM encryption scheme that is secure in the standard model. In both the cases of key-cycles and leakage of partial information, we would like to extend our results from the passive-adversary setting to that of the active adversary.

Also, one might consider various expansions of the formal setting that would allow additional operations such as *xor*, pseudorandom permutations, or exponentiation. Soundness and completeness such richer formal settings would, of course, need exploration. In particular, the definition of patterns appears to be rather subtle in such richer settings. We would also like to understand how our methods fit with the methods of [Mau02].

Lastly, one might consider exploring partial leakage in the setting of asymmetric encryption. One might also extend our methods and investigate formal treatment of other cryptographic primitives. It would be interesting to see if our methods could be combined with the methods of [BPW03, Can01].

As for Chapter 3 we would like to extend the expressiveness of our calculus. It would be interesting to extend it with secrecy and probabilistic choices. Also, it would be interesting to see if the techniques developed to prove the soundness and completeness results for our calculus can be also applied to similar results for a full-fledge process calculus with explicit cryptography. Similarly to what was done in Chapter 2, we are also interested in extending our calculus to incorporate the notion of length of messages in the terms of the calculus. We envisage its implementation using types and sorts.

As for Chapter 4, one could try to extend the existent results for classical process algebras to the quantum world. For instance, one can assume that adversaries have quantum polynomial time power and try to find a sound and complete implementation of a process algebra over such quantum computational model as we did in Chapter 2 for the classical model.

Bibliography

[AB05] Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. *Journal of the ACM*, 52(1):102–146, January 2005.

[Aba98] Martín Abadi. Protection in programming-language translations. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 868–883. Springer, 1998. Also Digital Equipment Corporation Systems Research Center report No. 154, April 1998.

[Aba99] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, September 1999.

[ABHS05] P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In S. De Capitani di Vimercati, P. Syverson, and D. Gollmann, editors, *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396, Milan, Italy, September 12–14 2005. Springer.

[ABS05] P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In CSFW05 [CSF05], pages 170–184.

[AF01] M. Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages (POPL)*, pages 104–115, 2001.

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004. Special issue on Foundations of Wide Area Network Computing. Parts of this work were presented at PET'02 (LNCS 2482) and ISSS'02 (LNCS 2602).

[AF06a] P. Adão and C. Fournet. Cryptographically sound implementations for communicating processes. Technical Report MSR-TR-2006-49, Microsoft Research, 2006.

[AF06b] P. Adão and C. Fournet. Cryptographically sound implementations for communicating processes. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, Venice, Italy, July 9–16 2006. Springer. To Appear. Full version available as Microsoft Research Technical Report MSR-TR-2006-49, April 2006.

[AFG00] Martín Abadi, Cédric Fournet, and Georges Gonthier. Authentication primitives and their compilation. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages (POPL 2000)*, pages 302–315. ACM, 2000.

[AFG02] Martín Abadi, Cédric Fournet, and Georges Gonthier. Secure implementation of channel abstractions. *Information and Computation*, 174(1):37–83, 2002.

[AG97] M. Abadi and A. Gordon. Reasoning about cryptographic protocols in the Spi Calculus. In A. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the CONCUR'97 – 8^{th} International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 59–73, Warsaw, Poland, July 1997. Springer-Verlag.

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, Winter 1998.

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, January 1999. Full version available as SRC Research Report 149, January 1998.

[AJ01] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In N. Kobayashi and B. C. Pierce, editors, *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94, Sendai, Japan, October 29–31 2001. Springer.

[AM] P. Adão and P. Mateus. A process algebra for reasoning about quantum security. *Electronic Notes in Theoretical Computer Science*. To Appear. Preliminary version presented at QPL'05.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *Proceedings of the 1st IFIP International Conference on Theoretical Computer Science (IFIP TCS)*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22, Sendai, Japan, August 17–19 2000. Springer.

[AR02] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, January 2002. Preliminary version presented at IFIP TCS'00.

[AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the 10^{th} ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991.

[BAF05] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *Proceedings of the 20th IEEE Symposium on Logic in Computer Science (LICS)*, pages 331–340, Chicago, IL, USA, June 2005. IEEE Computer Society.

[BAN96] M. Burrows, M. Abadi, and R. Needham. A logic of authentication, from proceedings of the royal society, pp 233–271, volume 426, number 1871, 1989. In W. Stallings, editor, *Practical Cryptography for Data Internetworks*. IEEE Computer Society Press, 1996. A preliminary version appeared as Research Report 39, DEC Systems Research Center, Palo Alto, February 1989.

[Ban04] G. Bana. *Soundness and Completeness of Formal Logics of Symmetric Encryption*. PhD thesis, University of Pennsylvania, July 2004. Available at IACR ePrint Archive, Report 2005/101, April 2005.

[BB84] C.H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers Systems and Signal Processing*, pages 175–179, Bangalore, India, December, 10-12 1984. IEEE Computer Society Press.

[BBD+05] Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static validation of security protocols. *Journal of Computer Security*, 13(3):347–390, 2005.

[BD05] M. Backes and M. Dürmuth. A cryptographically sound Dolev-Yao style security proof of an electronic payment system. In CSFW05 [CSF05], pages 78–93.

[BDNN01] Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for the pi-calculus with applications to security. *Information and Computation*, 168(1):68–92, 2001.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In Krawczyk [Kra98], pages 26–45. Full version available at `http://www.cs.ucsd.edu/users/mihir/papers/relations.html`.

[Bea91] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[BGH+95] M. Bellare, J. Garay, R. Hauser, A. Herzberg abd H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP — a family of secure electronic payment protocols. In *Proceedings of the 1st USENIX Workshop on Electronic Commerce*, 1995.

[BGH+00] M. Bellare, J. Garay, R. Hauser, A. Herzberg abd H. Krawczyk, M. Steiner, G. Tsudik, E. Van Herreveghen, and M. Waidner. Design, implementation, and deployment of the ikp secure electronic payment system. *IEEE Journal on Selected Areas in Communications*, 18(4):611–627, 2000.

[BI03] M. Backes and C. Jacobi II. Cryptographically sound and machine-assisted verification of security protocols. In H. Alt and M. Habib, editors, *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686, Berlin, Germany, February 27–March 1 2003. Springer.

[Bla] B. Blanchet. A computationally sound mechanized prover for security protocols. Available at IACR ePrint Archive, Report 2005/401, November 2005.

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 11–13 2001. IEEE Computer Society Press.

[Bla06] B. Blanchet. A computationally sound mechanized prover for security protocols. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P)*, Oakland, CA, USA, 2006. IEEE Computer Society Press. To Appear. Full version available at IACR ePrint Archive, Report 2005/401, November 2005.

[BMV05] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model-checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[Bon03] D. Boneh, editor. *Advances in Cryptology - CRYPTO 2003 : 23rd Annual International Cryptology Conference Santa Barbara, California, USA, August 17–21, 2003 Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 17–21 2003. Springer.

[BP04a] M. Backes and B. Pfitzmann. A cryptographically sound security proof of the Needham-Schröeder-Lowe public-key protocol. *IEEE Journal on Selected Areas in Communications*, 22(10):2075–2086, December 2004. Preliminary version presented at FSTTCS'03.

[BP04b] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 204–218, Pacific Grove, CA, USA, June 28–30 2004. IEEE Computer Society Press. Full version available at IACR ePrint Archive, Report 2004/059.

[BP05] M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. *IEEE Transactions on Dependable and Secure Computing*, 2(2):109–123, April 2005. Preliminary version presented at S&P'05.

[BPS] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks. Available at IACR ePrint Archive, Report 2005/421, November 2005.

[BPW] M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. Available at IACR ePrint Archive, Report 2004/082.

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pages 220–230, Washington D.C., USA, October 27–30 2003. ACM Press. Full version available at IACR ePrint Archive, Report 2003/015, January 2003.

[BPW05] M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. *International Journal of Information Security*, 4(3):135–154, June 2005. Preliminary version presented at ESORICS'03.

[Bra99] S. Brands. Electronic cash. In M. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 44. CRC Press, November 1999.

[BRS02] J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In K. Nyberg and H. Heys, editors, *Proceedings of the 9th Annual International Workshop on Selected Areas in Cryptography (SAC)*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75, St. John's, Newfoundland, Canada, August 15–16 2002. Springer.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, Las Vegas, NV, USA, October 14–17 2001. IEEE Computer Society Press. Full version available at IACR ePrint Archive, Report 2000/067.

[CGG05] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Secrecy and group creation. *Information and Computation*, 196(2):127–155, 2005.

[CH06] R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In *Proceedings of the 3nd Theory of Cryptography Conference (TCC)*, Lecture Notes in Computer Science, pages 380–403, New York, NY, USA, March 5–7 2006. Springer. Full version available at IACR ePrint Archive, Report 2004/334.

[CKS01] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *Proceedings of the 8^{th} ACM Conference on Computer and Communications Security*, Philadelphia, Pennsylvania, November 2001. ACM Press.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 98–118, Innsbruck, Austria, May 6–10 2001. Springer.

[CMP05] I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In CSFW05 [CSF05], pages 48–61.

[CR73] S. A. Cook and R. A. Reckhow. Time bounded random access machines. *Journal of Computer and System Sciences*, 7(4):354–375, August 1973.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [Kra98], pages 13–25.

[CSF05] *Computer Security Foundations, 2005. CSFW-18 2005. 18th IEEE Workshop*, Aix-en-Provence, France, June 20–22 2005. IEEE Computer Society Press.

[CTDL77] C. Cohen-Tannoudji, B. Diu, and F. Laloë. *Quantum Mechanics*. John Wiley, 1977.

[CVB04] C. Caleiro, L. Viganò, and D. Basin. Towards a metalogic for security protocol analysis. In W. A. Carnielli, F. M. Dionísio, and P. Mateus, editors, *Proceedings of CombLog'04, Workshop on Combination of Logics: Theory and Applications*, pages 187–196, 1049-001 Lisboa, Portugal, 2004. Departamento de Matemática, Instituto Superior Técnico. Extended abstract.

[CVB05] C. Caleiro, L. Viganò, and D. Basin. Metareasoning about security protocols using distributed temporal logic. *Electronic Notes in Theoretical Computer Science*, 125(1):67–89, 2005. Preliminary version presented at IJCAR'04 ARSPA Workshop.

[CW05] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In Sagiv [Sag05], pages 157–171.

[DDM$^+$05] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the The 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29, Lisbon, Portugal, July 11–15 2005. Springer.

[DKMR05] A. Datta, R. Küsters, J. C. Mitchell, and A. Ramanathan. On the relationships between notions of simulation-based security. In Kilian [Kil05], pages 476–494.

[DY83] D. Dolev and A. C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983. Preliminary version presented at FOCS'81.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[Fei91] J. Feigenbaum, editor. *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11–15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 11–15 1991. Springer.

[FHG98] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings 1998 IEEE Symposium on Security and Privacy*, pages 160–171. Springer Verlag, May 1998.

[FHG99] F. J. Thayer Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. Practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[GJ03] Andrew D. Gordon and Alan Jeffrey. Authenticity by typing for security protocols. *Journal of Computer Security*, 11(4):451–520, 2003.

[GJ04] Andrew D. Gordon and Alan Jeffrey. Types and effects for asymmetric cryptographic protocols. *Journal of Computer Security*, 12(3-4):435–483, 2004.

[GL91] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93, Santa Barbara, CA, USA, August 11–15 1991. Springer.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28(2):270–299, April 1984. Preliminary version presented at STOC'82.

[GM95] J. Gray and J. McLean. Using temporal logic to specify and verify cryptographic protocols (progress report). In *Proceedings of the 8^{th} IEEE Computer Security Foundations Workshop (CSFW)*, pages 108–116. IEEE Computer Society Press, 1995.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GN05] S. J. Gay and R. Nagarajan. Communicating quantum processes. In J. Palsberg and M. Abadi, editors, *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 145–157, Long Beach, CA, USA, January 12–14 2005. ACM Press.

[GTZ01] J. D. Guttman, F. J. Thayer, and L. D. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 186–195, Philadelphia, PA, USA, November 05–08 2001. ACM Press.

[Her02] J. Herzog. Computational Soundness of Formal Adversaries. Master thesis, Massachussets Institute of Technology, 2002.

[Her04] J. Herzog. *Computational Soundness for Standard Assumptions of Formal Cryptography*. PhD thesis, Massachussets Institute of Technology, May 2004. Available at `http://theory.lcs.mit.edu/~jherzog/papers/herzog-phd.pdf`.

[HG03] O. Horvitz and V. Gligor. Weak key authenticity and the computational completeness of formal encryption. In Boneh [Bon03], pages 530–547.

[HLM03] J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In Boneh [Bon03], pages 548–564.

[Hoa80] C. A. R. Hoare. Communicating sequential processes. In R. McKeag and A. Macnaghten, editors, *On the construction of programs – an advanced course*, pages 229–254. Cambridge University Press, 1980.

[HVY00] Kohei Honda, Vasco Thudichum Vasconcelos, and Nobuko Yoshida. Secure information flow as typed process behaviour. In Gert Smolka, editor, *ESOP*, volume 1782 of *Lecture Notes in Computer Science*, pages 180–199. Springer, 2000.

[IK03] R. Impagliazzo and B. M. Kapron. Logics for reasoning about cryptographic constructions. In *44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 372–383, Cambridge, MA, USA, October 11–14 2003. IEEE Computer Society Press.

[JLM05] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In Sagiv [Sag05], pages 172–185.

[Kil05] J. Kilian, editor. *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10–12, 2005. Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, Cambridge, MA, USA, February 10–12 2005. Springer.

[Kni96] E. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.

[Kra98] H. Krawczyk, editor. *Advances in Cryptology - CRYPTO'98: 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23–27, 1998. Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 23–27 1998. Springer.

[Lau02] P. Laud. Encryption cycles and two views of cryptography. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NORDSEC)*, number 31 in Karlstad University Studies, pages 85–100, Karlstad, Sweden, November 7–8 2002.

[Lau04] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P)*, pages 71–85, Oakland, CA, USA, May 9–12 2004. IEEE Computer Society Press.

[Lau05] P. Laud. Secrecy types for a simulatable cryptographic library. In V. Atluri, C. Meadows, and A. Juels, editors, *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, pages 26–35, Alexandria, VA, USA, Nvember 7–11 2005. ACM Press.

[LC03] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In J. I. Lim and D. H. Lee, editors, *Proceedings of the 6th International Conference on Information Security and Cryptology (ICISC)*, volume 2971 of *Lecture Notes in Computer Science*, pages 55–66, Seoul, Korea, November 27–28 2003. Springer.

[LMMS98] P. Lincoln, J. C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polynomial-time framework for protocol analysis. In M. Reiter, editor, *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS)*, pages 112–121, San Francisco, CA, USA, November 3–5 1998. ACM Press.

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

[Low96] G. Lowe. Breaking and fixing the Needham-Shroeder public-key protocol using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *Proceedings of the 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[Mau02] U. Maurer. Indistinguishability of random systems. In L. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132, Amsterdam, The Netherlands, April 28–May 2 2002. Springer.

[Mea92] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1):5–36, 1992.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil99] R. Milner. *Communicating and Mobile Systems : The pi-Calculus*. Cambridge University Press, June 1999.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murφ. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy (S&P)*, pages 141–151, Oakland, CA, USA, May 4–7 1997. IEEE Computer Society Press.

[MMS03] P. Mateus, J. C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In R. Amadio and D. Lugiez, editors, *CONCUR 2003 - Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 327–349, Marseille, France, September 3–5 2003. Springer.

[MMS05] A. M. Martins, P. Mateus, and A. Sernadas. Minimization of quantum automata. Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2005. Submitted for Publication.

[MP05] D. Micciancio and S. Panjwani. Adaptive security of symbolic encryption. In Kilian [Kil05], pages 169–187.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–40, 41–77, September 1992.

[MR91] S. Micali and P. Rogaway. Secure computation. In Feigenbaum [Fei91], pages 392–404.

[MRS98] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1998.

[MRST01] J. C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols (preliminary report). *Electronic Notes in Theoretical Computer Science*, 45:1–31, November 2001. Preliminary version presented at MFPS'01.

[MRST04] J. C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In I. Walukiewicz, editor, *Proceedings of the 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2987 of *Lecture Notes in Computer Science*, pages 468–483, Barcelona, Spain, March 29–April 2 2004. Springer.

[MRST06] J. C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols. *Theoretical Computer Science*, 353(1–3):118–164, March 2006. Preliminary version presented at FOSSACS'04.

[MS04] P. Mateus and A. Sernadas. Reasoning about quantum systems. In J. J. Alferes and J. A. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 239–251, Lisbon, Portugal, September 27–30 2004. Springer.

[MS06] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204(5):771–794, 2006. ArXiv math.LO/0503453.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MW04a] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004. Preliminary version presented at WITS'02.

[MW04b] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Proceedings of the 1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, February 19–21 2004. Springer.

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

[Pau97a] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.

[Pau97b] L. C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.

[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[PSW00] B. Pfitzmann, M. Schunter, and M. Waidner. Cryptographic security of reactive systems. *Electronic Notes in Theoretical Computer Science*, 32:59–77, 2000. Preliminary version presented at WSAIF'99.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In P. Samarati, editor, *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, Athens, Greece, November 01–04 2000. ACM Press. Extended version (with M. Shunter) available as IBM Research Report RZ 3206, 2000, `http://www.zurich.ibm.com/security/models`.

[PW01] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the 2001 IEEE*

Symposium on Security and Privacy (S&P), pages 184–200, Oakland, CA, USA, May 14–16 2001. IEEE Computer Society Press.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard, USA, 1981.

[RS91] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Feigenbaum [Fei91], pages 433–444.

[RSA78] R. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sag05] M. Sagiv, editor. *Programming Languages and Systems: 14th European Symposium on Programming, ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4–8, 2005. Proceedings*, volume 3444 of *Lecture Notes in Computer Science*, Edinburgh, UK, April 4–8 2005. Springer.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–553, New York, NY, USA, October 17–19 1999. IEEE Computer Society Press.

[SC00] P. Syverson and I. Cervesato. The logic of authentication protocols. In R. Focardi and R. Gorrieri, editors, *Proceedings of the FOSAD'2000, International School on Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 63–136, Bertinoro, Italy, September 2000. Springer.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing*, 26(5):1484–1509, 1997.

[SM93] P. Syverson and C. Meadows. A logical language for specifying cryptographic protocol requirements. In *Proceedings of the* 14^{th} *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177, Los Alamitos, CA, USA, May 1993. IEEE Computer Society Press.

[SP00] P. W. Shor and J. Preskill. Simple proof of security of the BB84 quantum key distribution protocol. *Physical Review Letters*, 85:441–444, July 2000.

[Sti95] D. Stinson. *Cryptography: Theory and Practice*. Discrete Mathematics and Its Applications. CRC Press, 1995.

[Syv91] P. Syverson. The use of logic in the analysis of cryptographic protocols. In T. Lunt and J. McLean, editors, *Proceedings IEEE Symposium on Research in Security and Privacy*, pages 156–170. IEEE Computer Society, 1991.

[War03] B. Warinschi. A computational analysis of the Needham-Schröeder-(Lowe) protocol. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW)*, pages 248–262, Pacific Grove, CA, USA, June 30–July 2 2003. IEEE Computer Society Press.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, Chicago, IL, USA, November 3–5 1982. IEEE Computer Society Press.

Appendix A

Cryptographic Definitions

In this chapter we present the summary of the cryptographic primitives used in this dissertation. Some of these were already presented on previous chapters. We also state and prove the cryptographic results used in the rest of this dissertation. The first security notion is the notion of *negligible function*.

Definition A.1 (Negligible Function). A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible*, written $f(n) \leq \operatorname{neg}(n)$, if for any $c > 0$ there exists n_c such that $n \geq n_c$ implies $f(n) \leq n^{-c}$.

The notion of *computational indistinguishability* was introduced by Goldwasser and Micali [GM84] and is the notion that we used throughout this dissertation when comparing two computational systems.

Definition A.2 (Computational Indistinguishability). Two families $\{D_\eta\}_{\eta\in\mathbb{N}}$ and $\{D'_\eta\}_{\eta\in\mathbb{N}}$, are *indistinguishable*, written $D_\eta \approx D'_\eta$, if for all PPT adversaries A,

$$\left|\Pr\left[d \longleftarrow D_\eta; \mathsf{A}(1^\eta, d) = 1\right] - \Pr\left[d \longleftarrow D'_\eta; \mathsf{A}(1^\eta, d) = 1\right]\right| \leq \operatorname{neg}(\eta)$$

A.1 Cryptographic Primitives

An encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ with key generation $\mathcal{K}$, encryption $\mathcal{E}$ and decryption $\mathcal{D}$. Let $\mathbf{plaintexts}$, $\mathbf{ciphertexts}$, $\mathbf{publickey}$ and $\mathbf{secretkey}$ be nonempty subsets of $\mathbf{strings}$. The set $\mathbf{coins}$ is some probability field that stands for coin-tossing, *i.e.*, randomness.

Definition A.3 (Asymmetric Encryption Scheme). An asymmetric encryption scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{K} : \mathbb{N} \times \mathbf{coins} \rightarrow \mathbf{publickey} \times \mathbf{secretkey}$ is a key-generation algorithm with security parameter η,
- $\mathcal{E} : \mathbf{publickey} \times \mathbf{plaintexts} \times \mathbf{coins} \rightarrow \mathbf{ciphertexts}$ is an encryption function,

- $\mathcal{D}$: $\mathbf{secretkey} \times \mathbf{strings} \to \mathbf{plaintexts}$ is such that for all $(e, d) \in \mathbf{publickey} \times \mathbf{secretkey}$ and $\omega \in \mathbf{coins}$, $\mathcal{D}(d, \mathcal{E}(e, m, \omega)) = m$ for all $m \in \mathbf{plaintexts}$.

All these algorithms must be computable in polynomial-time in the size of the input. We insist that $\lceil \mathcal{E}(e, m, w) \rceil = \lceil \mathcal{E}(e, m, w') \rceil$ for all $e \in \mathbf{publickey}, m \in \mathbf{plaintexts}$ and $w, w' \in \mathbf{coins}$, where $\lceil x \rceil$ stands for the binary length of x.

There are several different notions of security for an encryption scheme. The one that we adopt here, introduced by [RS91], has been shown to be strictly stronger than almost all other definitions, including semantic security [BDPR98].

In the following, $\Pr[A; B : C]$ is the probability of occurrence of event C after performing events A and B.

Definition A.4 (IND-CCA2—Adaptive Chosen Ciphertext Security). A public-key encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ provides indistinguishability under the adaptive chosen-ciphertext attack if for all PPT adversaries A:

$$\begin{array}{lll} \Pr[& (e, d) \longleftarrow \mathcal{K}(1^\eta); & \\ & m_0, m_1 \longleftarrow \mathsf{A}^{\mathcal{D}_1(\cdot)}(1^\eta, e); & \\ & b \longleftarrow \{0, 1\}\,; & \\ & c \longleftarrow \mathcal{E}(e, m_b); & \\ & g \longleftarrow \mathsf{A}^{\mathcal{D}_2(\cdot)}(1^\eta, e, c) : & \\ & b = g &] \leq \frac{1}{2} + \operatorname{neg}(\eta) \end{array}$$

The oracle $\mathcal{D}_1(x)$ returns $\mathcal{D}(d, x)$, and $\mathcal{D}_2(x)$ returns $\mathcal{D}(d, x)$ if $x \neq c$ and returns $\perp$ otherwise. The adversary is assumed to keep state between the two invocations. It is required that m_0 and m_1 be of the same length.

That is, an adversary should not be able to learn from a ciphertext whether it is the encryption of the plaintext m_0 or the plaintext m_1, even if the adversary knows the public key used to encrypt, the adversary can choose the messages m_0 and m_1 itself, so long as the messages have the same length, and the adversary can request and receive the decryption of any *other* ciphertext.

Definition A.5 (Signature Scheme). A signature scheme is a triple $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ where:

- $\mathcal{G}$: $\mathbb{N} \times \mathbf{coins} \to \mathbf{sigkey} \times \mathbf{verifykey}$ is a key-generation algorithm with security parameter η,

- $\mathcal{S}$: $\mathbf{sigkey} \times \mathbf{plaintexts} \times \mathbf{coins} \to \mathbf{signedtexts}$ is the signing algorithm, and

- $\mathcal{V}$: $\mathbf{verifykey} \times \mathbf{strings} \times \mathbf{signedtexts} \to \{0, 1\}$ is such that for every pair $(s, v) \longleftarrow \mathcal{G}(1^\eta)$ and $\omega \in \mathbf{coins}$, $\mathcal{V}(v, m, \mathcal{S}(s, m, \omega)) = 1$ for all $m \in \mathbf{plaintexts}$.

All these algorithms must be computable in polynomial-time in the size of the input. We call (s, v) a pair of signing/verification keys, and the string $\mathcal{S}(s, m, \omega)$ a signature of the plaintext m with the key s.

Similarly to encryption, there are several different notions of security for signature schemes. We adopt the notion of *unforgeable signature under chosen message attack* [GMR88].

Definition A.6 (Adaptive Chosen Message Security). A signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is secure against forgery under adaptive chosen-message attack if for all PPT adversaries A:

$$\begin{array}{lll} \Pr[& (s,v) \longleftarrow \mathcal{G}(1^\eta); & \\ & (m, sig) \longleftarrow \mathsf{A}^{\mathcal{S}_1(\cdot)}(1^\eta, v) : & \\ & \mathcal{V}(v, m, sig) = 1 & \mid \quad m \notin \mathit{Queries} \quad] \leq \operatorname{neg}(\eta) \end{array}$$

The oracle $\mathcal{S}_1(x)$ returns $\mathcal{S}(s,x)$ and adds x to the set $\mathit{Queries}$.

This game intuitively says that, after requesting as many signatures as he wants from the signing oracle $\mathcal{S}_1$, an adversary cannot produce a pair (m, sig) such that sig is the signature of the message m. Of course this game is only fair if the produced pair is not one of the pairs obtained by querying the oracle. Note that the adversary can also access the verification algorithm since he knows the verification key v.

How is CMA Security used in Chapter 3

Definition A.6 does not state whether the adversary (or even the signer) is able to generate other values sig' such that $\mathcal{V}(v, m, sig') = 1$ given v, m, and sig (and even s for the signer). In practice, this ability may come from the underlying cryptographic algorithms, or simply from the lack of normalisation for signature values.

Conservatively, our high-level semantics presented in Chapter 3 assumes this is always possible, as specified in Rule (SysIn), whereas our low-level implementation does not rely on this ability. Still, for establishing some of our results (e.g. the existence of some adversary in completeness proofs), we need to be more specific. To this end, we then use a signature scheme $\Sigma' = (\mathcal{G}, \mathcal{S}, \mathcal{V}, \mathcal{M})$ such that $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ meets Definition A.6 and the fourth algorithm $\mathcal{M}$ is such that, if $\mathcal{V}(v, m, sig) = 1$, then the values $sig_n = \mathcal{M}(v, m, sig, 1^n)$ for $n < \eta$ are pairwise distinct and such that $\mathcal{V}(v, m, sig_n) = 1$.

It is straightforward (if not very useful) to build such a signature scheme from any given Σ, as follows: for signing, we concatenate η zeros to the signature value sig; for signature verification, we ignore the last η bits of the signature value; for producing other signature value, we increment the last bits of the signature value.

As for authentication, we use a symmetric signing scheme. A symmetric authentication scheme $\Lambda = (\mathcal{G}_\Lambda, \mathcal{A}, \mathcal{C})$ is defined in the same way as Definition A.5 except that $\mathcal{G}_\Lambda$ generates a single key, used both for signing and encryption, that verifies the equation $\mathcal{C}(k, m, \mathcal{A}(k, m, \omega)) = 1$ for all $m \in \mathbf{plaintexts}$. The notion of security is the same as in Definition A.6, except that the adversary is not given the verification key v (which is also the signing key).

A.2 Cryptographic Results

Proposition A.1. *Let $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a CMA-secure authentication scheme and n any function bounded above by a polynomial in the security parameter η. Then for all PPT adversaries A:*

$$\begin{aligned}\Pr[\ & (s_i, v_i)_{i=1}^n \longleftarrow \mathcal{G}(1^\eta); \\ & (m, sig) \longleftarrow \mathsf{A}^{\mathcal{S}_1(\cdot),\mathcal{S}_2(\cdot),\ldots,\mathcal{S}_n(\cdot)}(1^\eta, v_1, v_2, \ldots, v_n) : \\ & \mathcal{V}(v_i, m, sig) = 1 \text{ for some } 1 \le i \le n \qquad \mid \quad m \notin Queries_i \quad] \le \operatorname{neg}(\eta)\end{aligned}$$

where the oracles $\mathcal{S}_i(x)$ returns $\mathcal{S}(s_i, x)$ and adds x to the set $Queries_i$.

Proof. Suppose that there is an adversary A that is able to create such signature with non-negligible probability. We can then create an adversary $\mathsf{Adv}_{\mathsf{CMA}}$ that can break CMA-security, Definition A.6. Define ${\mathsf{Adv}_{\mathsf{CMA}}}^{\mathcal{F}(\cdot)}(1^\eta, v)$ as follows:

```
generate $(s_2, v_2), (s_3, v_3), \ldots, (s_n, v_n) \longleftarrow \mathcal{K}(1^\eta)$
call adversary $\mathsf{A}(1^\eta, v, v_2, v_3, \ldots, v_n)$
  on every call of A to $\mathcal{S}_1(x)$ return the result $\mathcal{F}(x)$
  on every call of A to $\mathcal{S}_i(x)$ return the result $\mathcal{S}(s_i, x)$
  if A outputs $(m, sig)$ then
    output $(m, sig)$
```

It is immediate to see that if there is an adversary A that is able to break the property stated in the proposition with non-negligible probability, then this adversary A can be used to construct adversary ${\mathsf{Adv}_{\mathsf{CMA}}}^{\mathcal{F}(\cdot)}$ above that breaks CMA-security. □

Proposition A.2 (Equivalent Notion of Negligible Function). *Let $f : \mathbb{N} \to \mathbb{R}$ be a function. $f(n)$ is negligible if and only if*

$$\forall q(n) \exists n_0 : n \ge n_0 \implies f(n) \le \frac{1}{q(n)}.$$

Proof. Straightforward from Definition A.1. □

Proposition A.3. *Let $f : \mathbb{N} \to \mathbb{R}$ be a* negligible *function and $p : \mathbb{N} \to \mathbb{R}$ a polynomial.*
We have $p(n) \times f(n)$ is a negligible function.

Proof. By definition, $p(n) \times f(n)$ is a negligible function if

$$\forall q(n) \exists n_0 : n \ge n_0 \implies p(n) \times f(n) \le \frac{1}{q(n)}.$$

Let $q(n)$ be a polynomial. Since $f(n)$ is negligible, we have that

$$\forall q'(n) \exists n_0 : n \ge n_0 \implies f(n) \le \frac{1}{q'(n)}$$

Pick $q'(n) = p(n) \times q(n)$, then

$$\begin{aligned}
\exists n_0 : n \geq n_0 \quad &\implies \quad f(n) \leq \frac{1}{p(n) \times q(n)} \\
&\implies \quad p(n) \times f(n) \leq \frac{p(n)}{p(n) \times q(n)} = \frac{1}{q(n)},
\end{aligned}$$

hence $p(n) \times f(n)$ is a negligible function. $\square$

Appendix B

Soundness and KDM Security: the case of Asymmetric Encryption

In this appendix, we re-examine KDM security and soundness in the presence of key-cycles, but this time we consider the setting of asymmetric encryption. The material of this appendix is almost entirely the same as that of Section 2.2 with the following differences:

- The set of keys are split into encryption (public) keys and decryption (private) keys,
- Type-0 encryption is replaced by chosen-ciphertext secure encryption (CCA-2 in the notation of [BDPR98]; found in Definition A.4).

Our main result is the same: (asymmetric) KDM-secure encryption provides type-3 soundness in the presence of key-cycles. In the public-key encryption setting, this result has many powerful implications. Many extensions of the Abadi and Rogaway result simply rely on soundness as a 'black-box' assumption, and are not themselves hindered by key-cycles. By removing the key-cycle restriction from the Abadi-Rogaway result, it is removed from these extensions as well.

Consider, for example, the non-malleability results of Herzog [Her04]. In this setting, the adversary does not wish to distinguish two expressions but to transform one expression M into another expression M'. The formal adversary has only a limited power to do this, and can only produce formal messages in a set called the *closure* of M (denoted $C[M]$). Soundness for this non-malleability property is that no computational adversary, given the interpretation of M, can produce the interpretation of an expression outside $C[M]$. As Herzog shows, this soundness for this non-malleability property is directly implied by soundness for indistinguishability of messages. Because we show the KDM security soundness for message indistinguishability, this result of Herzog shows that it also provides soundness for non-malleability properties as well.

For the rest of this section, we present those definitions and proofs that differ from their counterparts in Section 2.2. Because this section is independent of and parallel to that section, we will use the same notation and names found there.

Definition B.1 (Expressions). Let **Keys** $= \{K_1, K_2, K_3, ...\}$ be an infinite discrete set of symbols, called the set of encryption keys, and **Keys**$^{-1} = \{K_1^{-1}, K_2^{-1}, K_3^{-1}, ...\}$ the corresponding set of decryption keys. Let **Blocks** be a finite subset of $\{0,1\}^*$. We define the *set of expressions*, **Exp**, by the grammar:

$$\textbf{Exp} ::= \textbf{Keys} \quad | \quad \textbf{Keys}^{-1} \quad | \quad \textbf{Blocks} \quad | \quad (\textbf{Exp}, \textbf{Exp}) \quad | \quad \{\textbf{Exp}\}_{\textbf{Keys}}$$

We will denote by *Keys*(M) the set of all encryption keys occurring in M and by *Keys*$^{-1}(M)$ the set of decryption keys in M. Expressions of the form $\{N\}_K$ are called *encryption terms*.

Definition B.2 (Visible Subterms, Recoverable Decryption Keys). Let $vis\,(M) \subseteq$ **Exp**, the *visible subterms of* M, be the smallest set of expressions containing M such that:

1. $(N_1, N_2) \in vis\,(M) \implies N_1 \in vis\,(M)$ and $N_2 \in vis\,(M)$, and
2. $\{N\}_K \in vis\,(M)$ and $K^{-1} \in vis\,(M) \implies N \in vis\,(M)$.

Let *R-Keys*(M), the set of *recoverable decryption keys* in M, be $vis\,(M) \cap \textbf{Keys}^{-1}$.

Definition B.3 (Formal Length). We introduce a function symbol with fresh letter ℓ with the following identities:

- For all blocks B_1 and B_2, $\ell(B_1) = \ell(B_2)$ iff $|B_1| = |B_2|$,
- For all expression M and key-renaming function σ, $\ell(M) = \ell(M\sigma)$,
- If $\ell(M_1) = \ell(N_1)$, $\ell(M_2) = \ell(N_2)$ then $\ell((M_1, M_2)) = \ell((N_1, N_2))$, and
- If $\ell(M) = \ell(N)$, then for all K_i, $\ell(\{M\}_{K_i}) = \ell(\{N\}_{K_i})$.

Definition B.4 (Pattern). We define the *set of patterns*, **Pat**, by the grammar:

$$\textbf{Pat} ::= \textbf{Keys} \mid \textbf{Keys}^{-1} \mid \textbf{Blocks} \mid (\textbf{Pat}, \textbf{Pat}) \mid \{\textbf{Pat}\}_{\textbf{Keys}} \mid \Box_{\textbf{Keys}, \ell(\textbf{Exp})}$$

The pattern of an expression M, denoted by *pattern*(M), is derived from M by replacing each encryption term $\{M'\}_K \in vis\,(M)$ (where $K^{-1} \notin$ *R-Keys*(M)) by $\Box_{K,\ell(M')}$.

For two patterns P and Q, $P =_3 Q$ is defined the following way:

- If $P \in \textbf{Blocks} \cup \textbf{Keys} \cup \textbf{Keys}^{-1}$, then $P =_3 Q$ iff P and Q are identical.
- If P is of the form $\Box_{K,\ell(M')}$, then $P =_3 Q$ iff Q is of the form $\Box_{K,\ell(N')}$, and $\ell(M') = \ell(N')$ in the sense of Definition B.3.
- If P is of the form (P_1, P_2), then $P =_3 Q$ iff Q is of the form (Q_1, Q_2) where $P_1 =_3 Q_1$ and $P_2 =_3 Q_2$.
- If P is of the form $\{P'\}_K$, then $P =_3 Q$ iff Q is of the form $\{Q'\}_K$ where $P' =_3 Q'$.

Definition B.5 (Key-Renaming Function). A bijection σ : **Keys** $\to$ **Keys** is called a *key-renaming function*. For any expression (or pattern) M, $M\sigma$ denotes the expression (or pattern) obtained from M by replacing all occurrences of keys K in M by $\sigma(K)$ (including those occurrences as indices of $\Box$) and all occurrences of keys K^{-1} in M by $(\sigma(K))^{-1}$.

Definition B.6 (Equivalence of Expressions). We say that two expressions M and N are *equivalent*, denoted by $M \cong N$, if there exists a key-renaming function σ such that *pattern*$(M) =_3$ *pattern*$(N\sigma)$.

Definition B.7 (Key-Cycles). A formal message M contains a *key-cycle* if it contains encryption terms $\{M_1\}_{K_1}$, $\{M_2\}_{K_2}, \ldots, \{M_n\}_{K_n}$ (where $\{M_i\}_{K_i}$ denotes the encryption of the message M_i with the public key K_i) such that M_i contains the key necessary to decrypt $\{M_{i+1}\}_{K_{i+1}}$ and M_n contains the key necessary to decrypt $\{M_1\}_{K_1}$. In this case we say that we have a key-cycle of length n.

Definition B.8 (Asymmetric Encryption Scheme). A *computational asymmetric encryption scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{K} : \mathbb{N} \times \mathbf{coins} \rightarrow \mathsf{publickey} \times \mathsf{secretkey}$ is a key-generation algorithm with security parameter η,
- $\mathcal{E} : \mathsf{publickey} \times \mathbf{plaintexts} \times \mathbf{coins} \rightarrow \mathbf{ciphertexts}$ is an encryption function, and
- $\mathcal{D} : \mathsf{secretkey} \times \mathbf{strings} \rightarrow \mathbf{plaintexts}$ is such that for all $(e, d) \in \mathsf{publickey} \times \mathsf{secretkey}$ and $\omega \in \mathbf{coins}$

$$\mathcal{D}(d, \mathcal{E}(e, m, \omega)) = m \quad \text{for all } m \in \mathbf{plaintexts}.$$

All these algorithms must be computable in polynomial time in the size of the input not counting the **coins**. (For this reason, the set $\mathbb{N}$ is usually represented as 1^*.) We insist that $|\mathcal{E}(e, m, w)| = |\mathcal{E}(e, m, w')|$ for all $e \in \mathsf{publickey}, m \in \mathbf{plaintexts}$ and $w, w' \in \mathbf{coins}$, where $|x|$ stands for the binary length of x. We also insist that $0^* \subseteq \mathbf{plaintexts}$.

The CONVERT function for asymmetric encryption can be found in Figure B.1.

Theorem B.1. *CCA-2 security does not imply soundness. That is, if there exists an encryption scheme secure against the chosen-ciphertext attack, then there exists another encryption scheme which is secure against the chosen-ciphertext attack but does not provide soundness.*

Proof. This proof is exactly analogous to the proof of Theorem 2.1. This is shown via a simple counter-example. Assuming that there exists an encryption scheme secure against the chosen-ciphertext attack, we will use it to construct another scheme which is also secure against the chosen-ciphertext attack. However, we will show that this new scheme allows the adversary to distinguish one particular expression M from another particular expression N, even though $M \cong N$.

Let M be $\{K^{-1}\}_K$. Let N be the expression $\{K_1^{-1}\}_{K_2}$. Since these two expressions are equivalent, an encryption scheme that enforces soundness requires that the family of distributions

$$\{(e, d) \longleftarrow \mathcal{K}(1^\eta); c \longleftarrow \mathcal{E}(e, d) : c\}_{\eta \in \mathbb{N}}$$

be indistinguishable from the family of distributions

$$\{(e_1, d_1) \longleftarrow \mathcal{K}(1^\eta); (e_2, d_2) \longleftarrow \mathcal{K}(1^\eta); c \longleftarrow \mathcal{E}(e_1, d_2) : c\}_{\eta \in \mathbb{N}}.$$

However, this is not implied by Definition A.4. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a CCA-2 secure encryption scheme. Again, we assume that private keys and ciphertexts have different and easily distinguished formats. We construct a second CCA-2 secure encryption scheme $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ as follows:

```
**algorithm** INITIALIZE$(M)$
    **for** $K \in$ *Keys*$(M)$ **do** $(\tau(K), \tau(K^{-1})) \longleftarrow \mathcal{K}(1^\eta)$

**algorithm** CONVERT$(M)$
    **if** $M = K$ where $K \in$ **Keys then**
        **return** $\tau(K)$
    **if** $M = K^{-1}$ where $K \in$ **Keys**$^{-1}$ **then**
        **return** $\tau(K^{-1})$
    **if** $M = B$ where $B \in$ **Blocks then**
        **return** $B$
    **if** $M = (M_1, M_2)$ **then**
        $x \longleftarrow$ CONVET$(M_1)$
        $y \longleftarrow$ CONVERT$(M_2)$
        **return** $[x, y]$
    **if** $M = \{M_1\}_K$ **then**
        $x \longleftarrow$ CONVERT$(M_1)$
        $y \longleftarrow \mathcal{E}(\tau(K), x)$
        **return** $y$
    **if** $M = \square_{K,\ell(M')}$, **then**
        $y \longleftarrow \mathcal{E}(\tau(K), 0^{|\Phi_\eta(M')|})$
        **return** $y$
```

Figure B.1: Interpretation algorithm for asymmetric encryption

- Let $\mathcal{K}' = \mathcal{K}$,
- Let $\mathcal{E}'$ be the following algorithm:

$$\mathcal{E}'(e, m) = \begin{cases} d & \text{if } m = d \\ \mathcal{E}(e, m) & \text{otherwise} \end{cases} .$$

  For many encryption schemes, key-pairs are recognisable as such via number-theoretic properties. Even when this is not the case, the test that a pair (e, d) corresponds to a pair encryption/decryption key can be conducted via the sub-algorithm:

  – Select a random plaintext r;
  – Let $c \longleftarrow \mathcal{E}(e, r)$;
  – Let $p \longleftarrow \mathcal{D}(m, c)$;
  – Test whether $p = r$.

- Let $\mathcal{D}'$ be the following algorithm:

$$\mathcal{D}'(d, c) = \begin{cases} d & \text{if } c = d \\ \mathcal{D}(d, c) & \text{otherwise} \end{cases} .$$

The scheme Π' acts exactly like Π unless the encryption algorithm $\mathcal{E}'$ is called on a pair (e, m) where m (when used as a decryption key) can decrypt a random value encrypted with e. However, if such a value for m is easy to guess by the adversary, or easy to compute for a randomly generated public key e, then the scheme Π could not be CCA-2 secure. Thus, the new scheme Π' must also be CCA-2 secure. However, it does not guarantee indistinguishability for the two distributions above. The first distribution will output decryption key while the second outputs a ciphertext, and these two distributions are easily distinguished by form alone. $\square$

Since CCA-2 security implies a number of other definitions [BDPR98], we can easily conclude that these other definitions also do not imply soundness:

Corollary B.2. *Soundness is not implied by any of: NM-CCA-1 security, IND-CCA-1 security, NM-CPA security, or IND-CPA (semantic) security.*

Definition B.9 (Asymmetric KDM Security). Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme. Let the two oracles $\mathsf{Real}_{\mathbf{d}}$ and $\mathsf{Fake}_{\mathbf{d}}$ be defined as follows:

- Suppose that for a fixed security parameter $\eta \in \mathbb{N}$, a family of keys is given: $\{(e_i, d_i) \longleftarrow \mathcal{K}(1^\eta)\}_{i \in \mathbb{N}}$. The adversary can now query the oracles providing them with a pair (j, g), where $j \in \mathbb{N}$ and $g : \mathsf{secretkey}^\infty \to \{0,1\}^*$ is a constant length, deterministic function and $\mathbf{d}$ is defined as the sequence $\langle d_1, d_2, \ldots \rangle$:
  - The oracle $\mathsf{Real}_{\mathbf{d}}$ when receiving this input returns $c \longleftarrow \mathcal{E}(e_j, g(\mathbf{d}))$;
  - The oracle $\mathsf{Fake}_{\mathbf{d}}$ when receiving this same input returns $c \longleftarrow \mathcal{E}(e_j, 0^{|g(\mathbf{d})|})$.

We say that the encryption scheme is *KDM-secure* if for all PPT adversaries A:

$$\Pr\left[(\mathbf{e}, \mathbf{d}) \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathsf{Real}_{\mathbf{d}}}(1^\eta, \mathbf{e}) = 1\right] - \\ \Pr\left[(\mathbf{e}, \mathbf{d}) \longleftarrow \mathcal{K}(1^\eta) : \mathsf{A}^{\mathsf{Fake}_{\mathbf{d}}}(1^\eta, \mathbf{e}) = 1\right] \leq \operatorname{neg}(\eta)$$

Theorem B.3 (KDM Security Implies Soundness). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a computational encryption scheme. If Π is KDM-secure, then Π provides soundness.*

Proof. For an arbitrary key K, let $\iota(K)$ denote the index of K. For an expression M, a set of formal decryption keys S, and a function τ defined on $(\textbf{Keys} \cup \textbf{Keys}^{-1}) \setminus S$ such that $\tau|_{\textbf{Keys}}$ takes values in $\mathsf{publickey}$ and $\tau|_{\textbf{Keys}^{-1}}$ takes values in $\mathsf{secretkey}$, we define a function $f_{M,S,\tau} : \mathbf{coins}^{e(M)} \times \mathsf{secretkey}^\infty \to \mathbf{strings}$ (where $e(M)$ is the number of encryptions in M) inductively in the following way:

- For $M = B \in \textbf{Blocks}$, let $f_{B,S,\tau} : \mathsf{secretkey}^\infty \to \mathbf{strings}$ be defined as $f_{B,S,\tau}(\mathbf{d}) = B$;
- For $M = K \in \textbf{Keys}$, let $f_{K,S,\tau} : \mathsf{secretkey}^\infty \to \mathbf{strings}$ be defined as $f_{K,S,\tau}(\mathbf{d}) = \tau(K)$;
- For $M = K^{-1} \in \textbf{Keys}^{-1} \cap S$, let $f_{K^{-1},S,\tau} : \mathsf{secretkey}^\infty \to \mathbf{strings}$ be defined as $f_{K^{-1},S,\tau}(\mathbf{d}) = d_{\iota(K)}$;

- For $M = K^{-1} \in \mathbf{Keys}^{-1} \cap \overline{S}$, let $f_{K^{-1},S,\tau} : \mathsf{secretkey}^{\infty} \rightarrow \mathbf{strings}$ be defined as $f_{K^{-1},S,\tau}(\mathbf{d}) = \tau(K^{-1})$;
- For $M = (M_1, M_2)$, let $f_{(M_1,M_2),S,\tau} : \mathbf{coins}^{e(M_1)} \times \mathbf{coins}^{e(M_2)} \times \mathsf{secretkey}^{\infty} \rightarrow \mathbf{strings}$ be defined as $f_{(M_1,M_2),S,\tau}(\omega_{M_1}, \omega_{M_2}, \mathbf{d}) = [f_{M_1,S,\tau}(\omega_{M_1}, \mathbf{d}), f_{M_2,S,\tau}(\omega_{M_2}, \mathbf{d})]$;
- For $M = \{N\}_K$, let $f_{\{N\}_K,S,\tau} : \mathbf{coins} \times \mathbf{coins}^{e(N)} \times \mathsf{secretkey}^{\infty} \rightarrow \mathbf{strings}$ be defined as $f_{\{N\}_K,S,\tau}(\omega, \omega_N, \mathbf{d}) = \mathcal{E}(\tau(K), f_{N,S,\tau}(\omega_N, \mathbf{d}), \omega)$.

We first prove that $[\![M]\!]_\Phi \approx [\![\textit{pattern}(M)]\!]_\Phi$. Suppose that $[\![M]\!]_\Phi \not\approx [\![\textit{pattern}(M)]\!]_\Phi$, which means that there is an adversary A that distinguishes the two distributions, that is

$$\Pr[x \longleftarrow [\![M]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1] - \Pr[x \longleftarrow [\![\textit{pattern}(M)]\!]_{\Phi_\eta} : \mathsf{A}(1^\eta, x) = 1]$$

is a non-negligible function of η. We will show that this contradicts the fact that the system is KDM-secure. To this end, we construct an adversary that can distinguish between the oracles $\mathsf{Real}_{\mathbf{d}}$ and $\mathsf{Fake}_{\mathbf{d}}$. Let $\mathcal{F}$ denote either of these oracles. Let $\mathbf{e} \in \mathsf{publickey}^{\infty}$ be the array of public keys that $\mathcal{F}$ outputs. From now on, let $S = \mathbf{Keys}^{-1} \setminus \textit{R-Keys}(M)$, and if $K^{-1} \in S$, let then $\tau(K) = e_{\iota(K)}$. Consider now the following algorithm:

```
algorithm $\mathsf{A}^{\mathcal{F}}(1^\eta, \mathbf{e}, M)$
    for $K^{-1} \in \textit{R-Keys}(M)$ do $(\tau(K), \tau(K^{-1})) \longleftarrow \mathcal{K}(1^\eta)$
    $y \longleftarrow \mathrm{CONVERT2}_{\mathbf{e}}(M, M)$
    $b \longleftarrow \mathsf{A}(1^\eta, y)$
    return $b$

algorithm $\mathrm{CONVERT2}_{\mathbf{e}}(M', M)$ with $M' \sqsubseteq M$
    if $M' = K$ where $K \in \mathbf{Keys}$ then
        return $\tau(K)$
    if $M' = K^{-1}$ where $K^{-1} \in \textit{R-Keys}(M)$ then
        return $\tau(K^{-1})$
    if $M = B$ where $B \in \mathbf{Blocks}$ then
        return $B$
    if $M' = (M_1, M_2)$ then
        $x \longleftarrow \mathrm{CONVERT2}_{\mathbf{e}}(M_1, M)$
        $y \longleftarrow \mathrm{CONVERT2}_{\mathbf{e}}(M_2, M)$
        return $[x, y]$
    if $M' = \{M_1\}_K$ with $K^{-1} \in \textit{R-Keys}(M)$ then
        $x \longleftarrow \mathrm{CONVERT2}_{\mathbf{e}}(M_1, M)$
        $y \longleftarrow \mathcal{E}(\tau(K), x)$
        return $y$
    if $M' = \{M_1\}_K$ with $K^{-1} \notin \textit{R-Keys}(M)$ then
        $\omega \longleftarrow \mathbf{coins}^{e(M_1)}$
        $y \longleftarrow \mathcal{F}(\iota(K), f_{M_1,S,\tau}(\omega, .))$
        return $y$
```

This algorithm applies the distinguisher $\mathsf{A}(1^\eta, \cdot)$ on the distribution $[\![M]\!]_\Phi$ when $\mathcal{F}$ is $\mathsf{Real}_\mathbf{d}$, and the distribution of $[\![\textit{pattern}(M)]\!]_\Phi$ when $\mathcal{F}$ is $\mathsf{Fake}_\mathbf{d}$. So, if $\mathsf{A}(1^\eta, \cdot)$ can distinguish $[\![M]\!]_\Phi$ and $[\![\textit{pattern}(M)]\!]_\Phi$, then $\mathsf{A}^\mathcal{F}(1^\eta, \mathbf{e}, \cdot)$ can distinguish $\mathsf{Real}_\mathbf{d}$ and $\mathsf{Fake}_\mathbf{d}$. But we assumed that $\mathsf{Real}_\mathbf{d}$ and $\mathsf{Fake}_\mathbf{d}$ cannot be distinguished, so $[\![M]\!]_\Phi \approx [\![\textit{pattern}(M)]\!]_\Phi$.

In a similar manner, we can show that $[\![N]\!]_\Phi \approx [\![\textit{pattern}(N)]\!]_\Phi$. It is easy to see that $[\![\textit{pattern}(M)]\!]_\Phi = [\![\textit{pattern}(N)]\!]_\Phi$, because the two patterns differ only by key-renaming. Hence $[\![M]\!]_\Phi \approx [\![N]\!]_\Phi$. □

Corollary B.4. *CCA-2 security does not imply KDM-security. If there exists an encryption scheme secure against the chosen-ciphertext attack, there exists an encryption scheme which is secure against the chosen-ciphertext attack but not KDM-secure.*

Proof. Suppose that there exists a CCA-2 secure encryption scheme. By Theorem B.1 there is a CCA-2 secure scheme Π such that Π does not satisfy soundness. If all CCA-2 encryptions schemes are KDM-secure, then Π is as well. By Theorem B.3, this means that Π satisfies soundness—a contradiction. □

Theorem B.5. *KDM security does not imply NM-CPA security. That is, there is an encryption scheme that is KDM-secure, but not NM-CPA secure.*

Proof. This is easily seen by inspecting the KDM-secure encryption scheme given by Black *et al.* in the random oracle model [BRS02]. Let $\mathcal{F}$ be a trapdoor permutation generator. Then:

- $\mathcal{K} = \mathcal{F}$ produces pairs (f, f^{-1}) where f encodes a trapdoor permutation and f^{-1} encodes its inverse,
- The encryption algorithm $\mathcal{E}$, on input (f, M), selects a random bit-string r and returns the pair $(f(r), RO(r) \oplus M)$ (where RO is the random oracle),
- $\mathcal{D}$, on input $(f^{-1}, C = (c_1, c_2))$, returns $RO\left(f^{-1}(c_1)\right) \oplus c_2$.

This scheme is not NM-CPA secure: it is simple to change the ciphertext associated with a message M into the ciphertext of a related message. Note that an encryption of M provides confidentiality by essentially applying a random r as a one-time pad. Thus, changing a single bit of the (second component of a) ciphertext changes the same bit of the plaintext. That is, if $C = (f(r), RO(r) \oplus M)$ is an encryption of M, one can easily create $C' = \left(f(r), RO(r) \oplus \overline{M}\right)$ (where $\overline{M}$ is the bit-wise complement of M). C' decrypts to $\overline{M}$. Thus, this KDM-secure encryption scheme does not provide non-malleability of ciphertexts. □

Corollary B.6. *KDM security implies neither NM-CCA1 security nor CCA2 security.*

Proof. Suppose that KDM implies NM-CCA1. Since NM-CCA1 implies NM-CPA we have that KDM implies NM-CPA—contradicting Theorem B.5. Thus, KDM cannot imply NM-CCA1. Similarly for CCA2. □

We conclude our discussion on the relationships between different notions of security by showing that soundness does not imply IND-CPA:

Theorem B.7. *Soundness does not imply IND-CPA. That is, if there exists an encryption scheme that provides soundness, there exists a scheme which provides soundness but is not IND-CPA.*

Proof. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a sound encryption scheme. Let $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be the following. Let $\mathcal{K}' = \mathcal{K}$. Let $\mathcal{E}'$ do the same on an input of a pair of a public key and a plaintext (k, x) as $\mathcal{E}$ for all plaintext, except when x is the security parameter given by k, in which case $\mathcal{E}'$ outputs a fixed bit-string σ of the same length as $\mathcal{E}(k, x)$. $\mathcal{D}'$ is the corresponding modified decryption algorithm.

This encryption scheme is still sound, because the interpretation of any expression with respect to $\mathcal{E}$ is indistinguishable from the interpretation of this same expression with respect to $\mathcal{E}'$. The reason for this is the following: For each security parameter, there is only one string that is encrypted differently by $\mathcal{E}$ and $\mathcal{E}'$. Let Φ and Φ' denote the respective interpretations. For any K public or private key, $[\![K]\!]_\Phi = [\![K]\!]_{\Phi'}$ trivially, and also $[\![B]\!]_\Phi = [\![B]\!]_{\Phi'}$ for any block B. Moreover these interpretations hit the security parameter with negligible probability. Now, for any expression M, if $[\![M]\!]_\Phi \approx [\![M]\!]_{\Phi'}$ and $[\![M]\!]_{\Phi'}$ hits the security parameter with negligible probability, then $[\![\{M\}_K]\!]_\Phi \approx [\![\{M\}_K]\!]_{\Phi'}$, and $[\![\{M\}_K]\!]_{\Phi'}$ hits the security parameter with negligible probability. Similarly for pairing. Therefore, by induction, the two interpretations of a given expression are indistinguishable.

On the other hand, it is easy to see, that Π' is not IND-CPA secure, because an adversary who submits as candidate messages the security parameter and 0^η (that is, outputs $m_0 = 0^\eta$, $m_1 = 1^\eta$) will certainly be able to determine which of the two messages was encrypted. $\square$

All these statements are summarised in Figure B.2.

$$\begin{tikzcd}
\textbf{Plaintext Awereness} \arrow[dr] & & & \textbf{RCCA-2} \arrow[dl] & \\
 & \textbf{NM-CCA2} \arrow[r, Leftrightarrow] \arrow[dl] & \textbf{IND-CCA2} \arrow[rr, dashed, "Theorem\ B.1", "/" marking] \arrow[dr] & & \textbf{Soundness} \arrow[ddl, dashed, "Theorem\ B.7"', "\backslash" marking] \\
\textbf{NM-CCA1} \arrow[rrr] \arrow[d] & & & \textbf{IND-CCA1} \arrow[d] \arrow[dlll, "/\!/" marking] & \\
\textbf{NM-CPA} \arrow[rrr] \arrow[urrr] & & & \textbf{IND-CPA} \arrow[r, dotted, shift right, "BRS"', "/" marking] & \textbf{KDM} \arrow[l, dotted, shift right, "BRS"'] \arrow[uu, dashed, "Theorem\ B.3"'] \arrow[llll, dashed, bend left, "Theorem\ B.5", "/" marking]
\end{tikzcd}$$

(Contributions of this work are represented by dashed arrows.)

Figure B.2: Relation Among Different Security Notions

Appendix C

Proofs for Theorems 3.4 and 3.5

In this chapter, we prove Theorems 3.4 and 3.5, relying on a series of intermediate machines derived from $\mathsf{M}(S, \mathsf{D})$. We assume that all systems S are in normal form.

We start by showing that, for all "correct" interactions of an adversary A and machine $\mathsf{M}(S, \mathsf{D})$, e.g., interactions where the adversary do not create fake keys, or do not create fake signatures, if $\mathsf{M}(S, \mathsf{D})$ evolves to state M', then there is a high-level transition φ such that $S \xrightarrow{\varphi} S'$ and $\mathsf{M}' = \mathsf{M}(S', \mathsf{D}')$ for some shadow D' of S'. This only represents what happens with "correct" interactions and hence we should also show what happens in the case that things go wrong.

We then define machine $\overline{\mathsf{M}}(S, \mathsf{D})$, which behaves as machine $\mathsf{M}(S, \mathsf{D})$ except that it monitors (decoded) high-level transitions and raises a flag when it detects a transition not enabled by the high level semantics. Afterwards, we define the family of $\overline{\mathsf{N}}^{\tilde{n}}(S, \mathsf{D})$ machines that perform some encryptions of 0's instead of the original messages and in the limit do not perform any encryption or signature, $\overline{\mathsf{N}}(S, \mathsf{D})$ machine. We use $\overline{\mathsf{N}}^{\tilde{n}}(S, \mathsf{D})$ to prove secrecy for messages and $\overline{\mathsf{N}}(S, \mathsf{D})$ to reduce the problem of falsifying a key to the security of the signing primitive.

Having these two extra constructions, and knowing that in the case of "correct" interactions, evolutions of $\mathsf{M}(S, \mathsf{D})$ correspond to high-level transitions, we show that $\mathsf{M}(S, \mathsf{D}) \approx \overline{\mathsf{N}}(S, \mathsf{D})$. We also show that in the case of "incorrect" interactions, the failure flag is set to $true$. We finish by proving that this flag is set to $true$ only with negligible probability and hence, except with negligible probability, transitions of $\mathsf{M}(S, \mathsf{D})$ correspond to high-level transitions.

In both $\overline{\mathsf{N}}^{\tilde{n}}(S, \mathsf{D})$ and $\overline{\mathsf{N}}(S, \mathsf{D})$, the interface with the adversary is the same as for $\mathsf{M}(S, \mathsf{D})$, that is a wire $\mathbf{in}_a$ and $\mathbf{out}_a$ for every principal $a \in \mathcal{H}$.

C.1 Auxiliary Machines

Our first auxiliary machine has the same runs as $\mathsf{M}(S, \mathsf{D})$, but in addition it performs global (polynomial) checks that can detect runtime failures to implement the high level semantics. The checks detect, respectively, a fake message from $b \in \mathcal{H}$, a fake certificate in a message from $e \in \mathsf{Prin} \setminus \mathcal{H}$, and a clash in the representation of names (leading to a non-injective shadow). When a check fails, a flag is set on an additional wire.

Definition C.1 (Defensive Machine). Let $\overline{\mathsf{M}}(S, \mathsf{D})$ be $\mathsf{M}(S, \mathsf{D})$ of Definition 3.26 modified as follows:

1. After unmarshaling an input as message $\ulcorner M \urcorner^{\mathsf{D}', a}$ from u to $a \in \mathcal{H}$, with authentication key k, (D' is the shadow obtained from D after unmarshaling) and before calling the interpreter,
   (a) if $u \in \mathcal{H}$, check that S includes a message M/i with $\mathsf{D}.wire(i) = \_, k, \_, \_, \_$.
   (b) if $u \in \mathsf{Prin} \setminus \mathcal{H}$, check that S has an input transition (M).
2. After generating a bitstring s for name n during marshalling, check that $s \neq \mathsf{D}.name(m)$ for any m defined in $\mathsf{D}.name$.

Once a check fails, a flag is set on an additional wire; we then say that the run fails.

We write $\mathsf{A}[\overline{\mathsf{M}}(S, \mathsf{D})] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{M})$ for a particular run that completes without failure. We let $\overline{\mathsf{A}}$ range over PPT adversaries that can read the flag. Hence, we define a run of $\overline{\mathsf{A}}$ and $\overline{\mathsf{M}}$ as in Definition 3.1, except that $\overline{\mathsf{A}}$ may read the flag after every exchange at Step 4.

Definition C.2. Machines $\overline{\mathsf{M}}^0$ and $\overline{\mathsf{M}}^1$ are indistinguishable, written $\overline{\mathsf{M}}^0 \approx \overline{\mathsf{M}}^1$, when for every PPT adversary $\overline{\mathsf{A}}$, we have

$$|\Pr[1 \longleftarrow \overline{\mathsf{A}}[\overline{\mathsf{M}}^0]] - \Pr[1 \longleftarrow \overline{\mathsf{A}}[\overline{\mathsf{M}}^1]]| \leq \operatorname{neg}(\eta).$$

Next, we consider the machines that constitute our system as a single machine that runs all their components. This change has no observable effect (as the components for each machine are activated to process messages on different wires, with no direct interaction between machines). The main difference is that the resulting machine, as well as the intermediate machines used in the proof below, do not have a natural interpretation as a distributed implementation.

These machines are used to reduce the security of a particular run of our low-level communication protocol to the security of encryption, by selectively replacing its encrypted payload with a string of 0s. They are parameterised by an index n_a for each $a \in \mathcal{H}$ that indicates how many runs of the protocol that sends a message to a are thus modified. First we need to extend the notion of shadow.

Definition C.3 (Extended Shadow State). Let $S = \Phi \vdash \nu\widetilde{n}.C$ be a system such that the configuration $C = \prod_{a \in \mathcal{H}} a[P_a] \mid \prod_{i \in I} M/i$ is in normal form. An *extended shadow state* for S, written D, consists of the following data structure:

- $prin \in \mathsf{Prin} \rightarrow (\{0,1\}^\eta)^5$ is a function from $u \in \mathsf{Prin}$ to bitstrings id_u, e_u, v_u, d_u, s_u such that $u \rightarrow id_u$ is injective, and for every $u \in \mathcal{H}$, we have $(e_u, d_u) \longleftarrow \mathcal{K}(1^\eta)$, and $(v_u, s_u) \longleftarrow \mathcal{G}(1^\eta)$.

  The bitstrings id_u, e_u, v_u are public for all $u \in \mathsf{Prin}$; d_u and s_u are public if $u \in \mathsf{Prin} \backslash \mathcal{H}$.

- $name \in \mathsf{Name} \rightharpoonup \{0,1\}^\eta$ is a partial injective function defined at least on every name that occurs free in S, and names that occur in Φ, $\mathsf{D}.certval$ or $\mathsf{D}.wire$ for which $\mathsf{D}.wire(i) = (\_, \_, \_, \_, \mathit{false})$.

  The bitstring $name(m)$ is public for every name $m \notin \widetilde{n}$.

- ni is a family of partial injective functions $ni^a : \mathsf{Name} \rightharpoonup \{0,1\}^\eta$ for each $a \in \mathcal{H}$, defined at least for all names of P_a that are not locally-restricted.

- $certval$ is a partial function from certificates $u\{V\}_\ell$ to $s \in \{0,1\}^\eta$ defined at least on the certificates of Φ, $\mathsf{D}.wire$ such that $\mathsf{D}.wire(i) = (\_,\_,\_,\_,\mathit{false})$, and all certificates in S of the form $u\{V\}_\ell$ with $\ell \neq 0$. It is also defined for all the certificates in V such that $u\{V\}_\ell$ is defined in $certval$. $certval$ satisfies the following property: if $certval(u\{V\}_\ell) = s$, then $\mathcal{V}(v_u, [\![\ulcorner V \urcorner^{\mathsf{D},\mathcal{H}}]\!], s) = 1$.

  The bitstring $certval(V)$ is public when $V \in \mathcal{M}(\Phi)$ or V issued by $u \notin \mathcal{H}$.

- $wire$ is a partial function from indices i to (M, k, s, del, b) defined at least on I, where $M = a{:}b\langle V\rangle$ with $a, b \in \mathcal{H}$, and $del = 0$ if $i \in I$ and $del = 1$ otherwise. If $b = false$, the bitstrings s and k are the output and the authentication key produced by $\mathtt{send}_b$ on input $[\![\ulcorner V \urcorner^{\mathsf{D},\mathcal{H}}]\!]$. If $b = true$, the bitstrings s and k are the output and the authentication key produced by the modified $\mathtt{send}_b$ protocol of Definition C.5 on input M. (Intuitively, the boolean flag indicates whether 0s are encrypted instead of the message payload. The flag is set to true only when using step (1b) of Definition C.5.)

  The bitstrings s and del are public.

- $keycache$ is a function from $a \in \mathcal{H}$ to sets of bitstrings such that, if there exists an i with $wire(i) = (M, k, \_, 1, \_)$ with M to a, then $k \in keycache(a)$.

- $ms^{\mathsf{D}}(\eta)$ is a polynomial that sets the padding-size of the implementations of S.

Without loss of generality, we assume that all compliant principals use the same identifiers for representing the same names, that is, for all $a, b \in \mathcal{H}$ and all names n, we have $ni^a(n) = ni^b(n)$ when defined. (Formally, our machines use name identifiers only for comparisons and table lookups, so injective renamings on these identifiers do not affect their behaviour.) Also, for simplicity, we will use the term shadow when referring an extended shadow.

Definition C.4 ($\mathsf{N}_\mathcal{H}$ Machine). Let S be a system with shadow state D. We define $\mathsf{N}_\mathcal{H}$ machine as the collection of machines $\mathsf{N}_\mathcal{H}(S,\mathsf{D}) = (\mathsf{N}_{\mathcal{H},a}(S,\mathsf{D}))_{a\in\mathcal{H}}$ that share two common tables $signed$ and $names$ defined as

- $signed(\ulcorner a\{V\}_0 \urcorner^{\mathsf{D},\mathcal{H}}) = certval(a\{V\}_0)$ when defined ($a \in \mathcal{H}$);

- $names(ni^a(n)) = name(n)$, if $ni^a(n)$ is defined for some $a \in \mathcal{H}$,

and where each $\mathsf{N}_{\mathcal{H},a}(S,\mathsf{D})$ machine is defined as $\mathsf{M}_a(S,\mathsf{D})$ in Definition 3.26 with the following changes:

- $p_a = \ulcorner P_a \urcorner^{\mathsf{D},\mathcal{H}}$;

- uses $[\![\cdot]\!]$ and $parse(\cdot)$ as the marshaling and unmarshaling algorithms.

We define the defensive variant $\overline{\mathsf{N}}_\mathcal{H}(S,\mathsf{D})$ as $\overline{\mathsf{M}}(S,\mathsf{D})$ in Definition C.1 except that in Step 1, $\ulcorner M \urcorner^{\mathsf{D}',a}$ is replaced by $\ulcorner M \urcorner^{\mathsf{D}',\mathcal{H}}$.

Intuitively these shared tables allow all principals to know which names and certificates were created by honest principals. These tables are the result of merging all $signed_a$ and $names_a$ tables of the $\mathsf{M}_a(S, \mathsf{D})$ machines. In $\mathsf{N}_{\mathcal{H}}$, the internal representation of certificates that were issued by honest principals always have label $\mathtt{0}$. This can be seen as an extension of the notion of self-certificates. Since $\mathsf{N}_{\mathcal{H}}$ controls all the honest principals, a certificate is self-issued if it was generated by someone of the group. One should remark that whenever sending a message to the adversary, $\mathsf{N}_{\mathcal{H},b}$ is always able to insert the correct label in a certificate from a since it can find it in $signed$. One can easily show the following lemma:

Lemma C.1. *Let* D *be a shadow,* $a \in \mathcal{H}$*, and* V *a closed term such that* $\ulcorner V \urcorner^{\mathsf{D},a}$ *is defined.*
We have $[\![\ulcorner V \urcorner^{\mathsf{D},a}]\!]_a = [\![\ulcorner V \urcorner^{\mathsf{D},\mathcal{H}}]\!]$ *and* $[\![parse_a(s)]\!]_a = [\![parse(s)]\!]$.

Proof. The proof is by structural induction on V; it relies on the definitions of $[\![\cdot]\!]$, $parse(\cdot)$, $\ulcorner \cdot \urcorner^{\mathsf{D},X}$, and implementation of systems. □

We are now ready to define $\overline{\mathsf{N}}^{\widetilde{n}}$ machines.

Definition C.5 ($\overline{\mathsf{N}}^{\widetilde{n}}$ Machine). Let $\widetilde{n}$ be a function from $a \in \mathcal{H}$ to $n_a \in \mathbb{N} \cup \{\omega\}$ and S be a system with extended shadow D.

We extend the state of $\overline{\mathsf{N}}_{\mathcal{H}}(S, \mathsf{D})$ with a table $enctable$ that associates bitstrings msg to $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}, k$ for every $\mathsf{D}.wire(i) = M, k, id_a\text{-}id_b\text{-}msg, \_, true$. Intuitively, msg is decoded by table lookup instead of decryption. For each $a \in \mathcal{H}$, we let j_a be the number of messages to principal a recorded in $enctable$.

We define machine $\overline{\mathsf{N}}^{\widetilde{n}}(S, \mathsf{D})$ as $\overline{\mathsf{N}}_{\mathcal{H}}(S, \mathsf{D})$ of Definition C.4 modified as follows:

1. For each outgoing message M from a to $b \in \mathcal{H}$, if $j_b < n_b$ then
   (a) marshal the message content as in $\overline{\mathsf{N}}_{\mathcal{H}}(S, \mathsf{D})$ then discard the resulting bitstring;
   (b) use protocol $\mathtt{send}_b$ (Definition 3.22) modified as follows:
      - Instead of computing the message for encryption (Step 2), let $m = 0^{ms(\eta)}$, where $ms(\eta)$ is the size of marshaled messages for security parameter η.
      - After encrypting m into msg (Step 3), add entry $(msg, \ulcorner M \urcorner^{\mathsf{D}',\mathcal{H}}, k)$ to $enctable$, where D' is an extension of D.

   Otherwise proceed as $\overline{\mathsf{N}}_{\mathcal{H}}(S, \mathsf{D})$.

2. Use protocol $\mathtt{receive}_a$ (Definition 3.23) modified as follows:

   For each input bitstring $id_u\text{-}id_a\text{-}msg$, before decryption (Step 1), if $enctable$ associates msg to $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}, k$, then

   - check that M is from u to a, check that $k \notin keycache_a$, add k to $keycache_a$ (Step 4), and pass $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}$ to the interpreter (bypassing unmarshaling). If any of the checks fails, reject the message.

   Otherwise proceed as $\overline{\mathsf{N}}_{\mathcal{H}}(S, \mathsf{D})$.

We define two variants of $\overline{\mathsf{N}}^{\widetilde{\omega}}$ by modifying Step 1a above; these modifications remove the side effects of marshalling, that is, the generation of signature values and name representations recorded in $\mathsf{D}.certval$ and $\mathsf{D}.name$.

- $\overline{\mathsf{N}}^{\star}$ is $\overline{\mathsf{N}}^{\widetilde{\omega}}$ where marshalling in Step 1a does not marshal certificates;
- $\overline{\mathsf{N}}$ is $\overline{\mathsf{N}}^{\widetilde{\omega}}$ without marshalling at Step 1a.

C.2 Auxiliary Lemmas

In this section we state some auxiliary lemmas that we will use in the next sections.

Lemma C.2 (Correctness of Comparisons). *Let D be a shadow such that $\mathsf{D}.certval$ is injective, $a \in \mathcal{H}$, and V and V' two closed terms such that $\ulcorner V \urcorner^{\mathsf{D},a}$ and $\ulcorner V' \urcorner^{\mathsf{D},a}$ are defined.*

We have $V = V'$ if and only if $\ulcorner V \urcorner^{\mathsf{D},a} = \ulcorner V' \urcorner^{\mathsf{D},a}$.

Proof. The proof is by structural induction on V; it relies on the injectivity of $\mathsf{D}.certval$, $\mathsf{D}.ni^a$ and $\mathsf{D}.prin.id$. □

Lemma C.3 (Correctness of Pattern Matching). *Let D be a shadow such that $\mathsf{D}.certval$ is injective.*

Matching on internal representations of a with shadow D coincides with high-level matching.

Proof. Immediate from Lemma C.2. □

Lemma C.4. *Let S be a system where all local processes are stable except for principal a, and D a shadow for S such that $\mathsf{D}.certval$ is injective.*

If $\overline{\mathsf{M}}(S, \mathsf{D}) \rightsquigarrow^* \mathsf{M}'$, then there exist S' with shadow D', that differs from D at most in $\mathsf{D}'.ni^a$, and $S \rightarrow^* S'$ such that $\mathsf{M}' = \overline{\mathsf{M}}(S', \mathsf{D}')$, and $\mathsf{D}'.certval$ is injective.

Proof. This proof is done by induction in the length of the reduction, relying on the deterministic scheduling, and Lemma C.3. □

C.3 Partial Completeness for Low-Level Runs

In this section we show that, for any run with a given adversary, either our defensive machine correctly implements a series of high-level normal transitions, or it detects a runtime failure. These lemmas do not depend on probabilities. We also show that the probability of occurrence of a failure is negligible.

Before doing that, we need to set some notations. Given an arbitrary run $\mathsf{A}[\mathsf{M}(S, \mathsf{D})] \longrightarrow s_{\mathtt{r}}(\mathsf{M}')$, and since all the algorithms running within the local machines always terminate in polynomial time, except possibly for the interpreter, we can decompose any part of the run that involves one of these machines by assuming that the machine is interrupted just before every call to the scheduling function. Accordingly, we let low-level steps range over $\mathsf{M} \overset{(s)}{\rightsquigarrow}_a \mathsf{M}'$, $\mathsf{M} \rightsquigarrow_a \mathsf{M}'$, and $\mathsf{M} \overset{s}{\rightsquigarrow}_a \mathsf{M}'$ for a bitstring input, an internal interpretation step, and a bitstring output performed by machine M_a, respectively.

We also introduce a new rule for our labelled semantics of Section 3.2.2. In order to decompose our proofs into small low-level steps, we separate the high-level transitions in two stages. We let $S \xrightarrow{(\alpha)\bullet} S'$ be the auxiliary input transition defined by the base rule

$$(\textsc{CfgIn}\bullet)\frac{M \text{ from } c \in \mathsf{Prin} \text{ to } a \quad c \neq a}{a[P] \xrightarrow{(M)\bullet} a[M \mid P]}$$

plus all the context rules for inputs of Section 3.2.2. Intuitively, a transition labelled $(\alpha)\bullet$ represents the first stage of the input, which does not depend on the local process. With this high-level semantics, if $S \xrightarrow{(\alpha)\bullet} S^\bullet$, then either α is a previously intercepted message between honest principals, or α is constructed by the adversary, using only certificates built from Φ (Rule $\textsc{SysIn}$).

Let us now start with the proof of the completeness theorem for the case of a single low-level exchange.

Lemma C.5. *Let S be a safe stable system with shadow D such that $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective.*

For every bitstring inp, one of the following holds:

1. $\overline{\mathsf{M}}(S, \mathsf{D}) \overset{(inp)}{\rightsquigarrow}\rightsquigarrow^* \overset{s}{\rightsquigarrow} \mathsf{M}'$ *and there exist S' with shadow D' and normal transitions $S \xrightarrow{\alpha\widetilde{\beta}} S'$ such that $\mathsf{M}' = \overline{\mathsf{M}}(S', \mathsf{D}')$ and $\mathsf{D}'.name$ and $\mathsf{D}'.certval$ are injective; or*

2. $\overline{\mathsf{M}}(S, \mathsf{D}) \overset{(inp)}{\rightsquigarrow} \overset{\mathtt{done}}{\rightsquigarrow} \mathsf{M}'$ *and there exists D', shadow for S, such that $\mathsf{M}' = \overline{\mathsf{M}}(S, \mathsf{D}')$ and $\mathsf{D}'.name$ and $\mathsf{D}'.certval$ are injective; or*

3. $\overline{\mathsf{M}}(S, \mathsf{D}) \overset{(inp)}{\rightsquigarrow}\rightsquigarrow^* \overset{s}{\rightsquigarrow} \mathsf{M}'$ *and the failure flag is set to true in M'.*

Proof. Let inp be a bitstring that is given as an input to $\overline{\mathsf{M}}(S, \mathsf{D})$. Let us suppose without loss of generality that $inp = id_u\_id_a\_msg$, that is, it is addressed to principal a. When this message is received, it is dispatched to the $\mathtt{receive}_u$ protocol of a. Analysing $\mathtt{receive}_u$ protocol we have that:

- the $\mathtt{receive}_u$ protocol fails, i.e., one of the Steps 1–4 of Definition 3.23 fails, inp is discarded, and $\mathtt{done}$ is output by the machine. In this case, since there were no change in the state of $\overline{\mathsf{M}}(S, \mathsf{D})$, we are in Case 2 with $\mathsf{D}' = \mathsf{D}$; or

- the message is accepted, i.e., $s\_id_u\_k\_s_{sig}\_s_{auth} = \mathcal{D}(d_a, msg)$, all the checks in Steps 1–4 of Definition 3.23 succeed, k is added to $keycache_a$, and s is returned to the unmarshal function.

If the message is accepted by the $\mathtt{receive}_u$ protocol, then s is passed to the $parse_a(\cdot)$ function, Definition 3.21. In both cases of failure or success of $parse_a(s)$, we have that the internal state of the machine may have changed as new pairs (ind, s') may have been added to $names_a$.

- If $\mathit{parse}_a(s)$ fails, this message is discarded, and `done` is output by the machine. In this case, defining D' as D except that
  - $\mathsf{D}'.\mathit{keycache}(a) = \mathsf{D}.\mathit{keycache}(a) \cup \{k\}$,

  and for each new pair (ind, s') in names_a
  - $\mathsf{D}'.\mathit{ni}^a(n) = \mathit{ind}$ if there is a name n such that $\mathsf{D}.\mathit{name}(n) = s'$; or
  - $\mathsf{D}'.\mathit{name}(m) = s'$ and $\mathsf{D}'.\mathit{ni}^a(m) = \mathit{ind}$ for a fresh new abstract name m,

  we are in Case 2. $\mathsf{D}'.\mathit{name}$ is injective by construction, and $\mathsf{D}'.\mathit{certval} = \mathsf{D}.\mathit{certval}$ hence it is injective by hypothesis.

- If $\mathit{parse}_a(s)$ succeeds, defining $\mathsf{D}^\bullet$ as D except that
  - $\mathsf{D}^\bullet.\mathit{keycache}(a) = \mathsf{D}.\mathit{keycache}(a) \cup \{k\}$,

  for each new pair (ind, s') in names_a
  - $\mathsf{D}^\bullet.\mathit{ni}^a(n) = \mathit{ind}$ if there is a name n such that $\mathsf{D}.\mathit{name}(n) = s'$; or
  - $\mathsf{D}^\bullet.\mathit{name}(m) = s'$ and $\mathsf{D}^\bullet.\mathit{ni}^a(m) = \mathit{ind}$ for a fresh new abstract name m,

  and for each certificate $\texttt{cert\_}s_1\_s_2\_s_3$ that was successfully unmarshaled to $v_1\{v_2\}_{s'}$,
  - $\mathsf{D}^\bullet.\mathit{certval}(v_1\{v_2\}_\ell) = s'$ if there is no ℓ such that $\mathsf{D}.\mathit{certval}(v_1\{v_2\}_\ell) = s'$,

  we have that the message m passed to the interpreter is $\ulcorner M \urcorner^{\mathsf{D}^\bullet, a} = \ulcorner u{:}a\langle V\rangle \urcorner^{\mathsf{D}^\bullet, a}$, for some high-level term V.

If $\overline{\mathsf{M}}(S, \mathsf{D}) \overset{(\mathit{inp})}{\rightsquigarrow} \mathsf{M}^\bullet$ and the failure flag is set in $\mathsf{M}^\bullet$, we are in Case 3. If the flag is not set, we have by Definition C.1 that

1. if $u \in \mathcal{H}$, then S defines M/i and $\mathsf{D}.\mathit{wire}(i) = \_, k, \_, \_$; or
2. if $u \notin \mathcal{H}$, then S has an input transition (M).

In the first case consider $S \xrightarrow{(i)\bullet} S^\bullet$, and update $\mathsf{D}^\bullet$ modifying the value del of $\mathsf{D}^\bullet.\mathit{wire}(i)$ to 1. In the second case consider $S \xrightarrow{(M)\bullet} S^\bullet$. Notice that in this second case names and certificates that were private, do not become public. In either case there exist $S^\bullet$ with shadow $\mathsf{D}^\bullet$, and α such that $S \xrightarrow{\alpha\bullet} S^\bullet$, $\mathsf{M}^\bullet = \overline{\mathsf{M}}(S^\bullet, \mathsf{D}^\bullet)$, and both $\mathsf{D}^\bullet.\mathit{name}$ and $\mathsf{D}^\bullet.\mathit{certval}$ are injective by construction.

Suppose now that $\overline{\mathsf{M}}(S^\bullet, \mathsf{D}^\bullet) \rightsquigarrow^* \mathsf{M}^*$. Applying Lemma C.4 to $S^\bullet$, $\mathsf{D}^\bullet$, and M^* we have that there exist S^* with shadow D^* that differs from $\mathsf{D}^\bullet$ at most in $\mathsf{D}^*.\mathit{ni}^a$, and $S^\bullet \twoheadrightarrow S^*$ such that $\mathsf{M}^* = \overline{\mathsf{M}}(S^*, \mathsf{D}^*)$ and $\mathsf{D}^*.\mathit{certval}$ is injective. Moreover, the number of reduction steps is bound by $p_S(\lceil\alpha\rceil)$ since S is safe.

Suppose now that $\overline{\mathsf{M}}(S^*, \mathsf{D}^*) \overset{s}{\rightsquigarrow} \mathsf{M}'$. While marshaling, names_a and signed_a are updated, so let $\mathsf{D}' = \mathsf{D}^*$ except that for each new pair (ind, s') in names_a,

- $\mathsf{D}'.name(n) = s'$ for the name n such that $\mathsf{D}.ni^a(n) = ind$,

and for each new pair $(v_1\{v_2\}_{\mathsf{o}}, s')$ in $signed_a$,

- $\mathsf{D}'.certval(v_1\{v_2\}_{\mathsf{o}}) = s'$.

If the failure flag is set in M', we are in Case 3. If it is not, we have by Definition C.1 that $\mathsf{D}'.name$ is injective as the bitstrings generated for all ind do not coincide with any other bitstring in $\mathsf{D}^*.name$ Also, $\mathsf{D}'.certval$ is injective by construction. We have that $s = \ulcorner M_1 \urcorner^{\mathsf{D}',a} \dots \ulcorner M_m \urcorner^{\mathsf{D}',a}$ for some high level messages $M_1, \dots, M_m$. Let $M_{i_1} = a{:}u_1\langle V_1\rangle, \dots, M_{i_j} = a{:}u_j\langle V_j\rangle$ be the messages in s addressed to honest principals and $M_{i_{j+1}}, \dots, M_{i_m}$ the messages addressed to $e \notin \mathcal{H}$. We have that

$$S^* \xrightarrow{\nu i_1.a:u_1 \;\dots\; \nu i_j.a:u_j \; M_{i_{j+1}} \;\dots\; M_{i_m}} S'$$

where $S' = \Phi' \vdash S^+ \mid M_{i_i}/i_1 \mid \cdots \mid M_{i_j}/i_j$, S^+ is obtained from S^* by removing from P_a all the messages $M_1, \dots, M_n$, and $\Phi' = \Phi \cup \bigcup_{l=j+1}^{m} \phi(M_{i_l})_{\mathcal{H}}$. Extend D' such that for each M_{i_l} to honest principal u_l, $\mathsf{D}'.wire(i_l) = (M_{i_l}, k_l, s_{i_l}, 0, \textit{false})$, where k_l is the authentication key produced by $\mathtt{send}_b$ on input $[\![\ulcorner V_j \urcorner^{\mathsf{D}',a}]\!]$. We have that $\mathsf{M}' = \overline{\mathsf{M}}(S', \mathsf{D}')$.

Hence, we have shown that if $\overline{\mathsf{M}}(S, \mathsf{D}) \overset{(inp)}{\rightsquigarrow} \rightsquigarrow^* \overset{s}{\rightsquigarrow} \mathsf{M}'$ and the failure flag is not set in M', there exist S' with shadow D', $\mathsf{D}'.name$ and $\mathsf{D}'.certval$ injective, and normal transitions $S \xrightarrow{\alpha\tilde{\beta}} S'$ such that $\mathsf{M}' = \overline{\mathsf{M}}(S', \mathsf{D}')$. □

We now extend the result by induction to the case of multiple exchanges.

Lemma C.6. *Let S° be a safe initial system with initial shadow D° and A a PPT algorithm.*

If $\mathsf{A}[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{M})$, then there exist S with shadow D and normal transitions $S^\circ \xrightarrow{\varphi} S$ such that $\mathsf{M} = \overline{\mathsf{M}}(S, \mathsf{D})$ and $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective.

Proof. By Definition 3.1, the run $\mathsf{A}[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{M})$ can be decomposed as a series of n low-level exchanges

$$\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{M} \tag{C.1}$$

that do not set the failure flag. By induction on n, we show the existence of S with shadow D and normal transitions $S^\circ \xrightarrow{\varphi} S$ such that $\mathsf{M} = \overline{\mathsf{M}}(S, \mathsf{D})$ and $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective.

Base case $n = 0$: We have $\mathsf{M} = \overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$. We use $S = S^\circ, \mathsf{D} = \mathsf{D}^\circ$, and $\varphi = \varepsilon$. The partial function $\mathsf{D}^\circ.name$ is initially undefined for all names, hence injective, the same for $\mathsf{D}^\circ.certval$.

Inductive case: Suppose that

$$\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{M}_n \overset{(inp_{n+1})}{\rightsquigarrow} \rightsquigarrow^* \overset{s_{n+1}}{\rightsquigarrow} \mathsf{M}$$

and that the failure flag is not set in M. By induction hypothesis for n, there exist S' with shadow D' and normal transitions $S^\circ \xrightarrow{\varphi'} S'$ such that $\mathsf{M}_n = \overline{\mathsf{M}}(S', \mathsf{D}')$ and $\mathsf{D}'.name$ and $\mathsf{D}'.certval$ are injective. Moreover S' is safe (since S° is) and stable (by definition of normal transitions). By applying Lemma C.5 to S', D', and inp_{n+1}, the $n+1$-th exchange is described by one of the cases below:

1. $\overline{\mathsf{M}}(S', \mathsf{D}') \overset{(inp_{n+1})}{\rightsquigarrow}\rightsquigarrow^* \overset{s_{n+1}}{\rightsquigarrow}$ M and there exist S with shadow D and normal transitions $S' \xrightarrow{\alpha\widetilde{\beta}} S$ such that $\mathsf{M}' = \overline{\mathsf{M}}(S, \mathsf{D})$ and $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective.

   In this case , we conclude with S, D, and transitions with labels $\varphi = \varphi'\alpha\widetilde{\beta}$.

2. $\overline{\mathsf{M}}(S', \mathsf{D}') \overset{(inp_{n+1})\mathtt{done}}{\rightsquigarrow}$ M and there exists D, shadow for S', such that $\mathsf{M}' = \overline{\mathsf{M}}(S', \mathsf{D})$ and $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective.

   In this case, we conclude with $S = S'$, D, and transitions with labels $\varphi = \varphi'$.

3. $\overline{\mathsf{M}}(S', \mathsf{D}') \overset{(inp_{n+1})}{\rightsquigarrow}\rightsquigarrow^* \overset{s_{n+1}}{\rightsquigarrow}$ M and the failure flag is set to *true* in M'.

   This case is excluded by hypothesis.

□

Lemma C.7 (Partial Completeness for a Run of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$). *Let S° be a safe initial system with initial shadow D° and* A *a PPT algorithm.*

If $\mathsf{A}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N})$, *then there exist S with shadow* D *and normal transitions* $S^\circ \xrightarrow{\varphi} S$ *such that* $\mathsf{N} = \overline{\mathsf{N}}(S, \mathsf{D})$ *and* $\mathsf{D}.name$ *and* $\mathsf{D}.certval$ *injective.*

Proof Sketch. The proof is done by induction in the number of exchanges as the proof for Lemma C.6, using a modified version of Lemma C.5. We briefly sketch the proof pointing out the differences.

Suppose that $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \ldots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \overline{\mathsf{N}}(S_n, \mathsf{D}_n) \overset{(inp)}{\rightsquigarrow}\rightsquigarrow^* \overset{s}{\rightsquigarrow}$ N and the failure flag is not set in N. We want to show that either

1. there exist S with shadow D and normal transitions $S_n \xrightarrow{\alpha\widetilde{\beta}} S$ such that $\mathsf{N} = \overline{\mathsf{N}}(S, \mathsf{D})$; or

2. there exists D shadow for S such that $s = \mathsf{done}$ and $\mathsf{N} = \overline{\mathsf{N}}(S_n, \mathsf{D})$.

We should start by pointing out that all messages that are not in $enctable$ will be processed by `receive` the same way as in Lemma C.5. As for $parse(\cdot)$ we just have to notice that in the case of $\overline{\mathsf{N}}$ machines we use the same internal representation for names and certificates for all principals, that is, there is a common identifier ind such that $ind = \mathsf{D}_n.ni(n) = \mathsf{D}_n.ni^a(n)$ for all $a \in \mathcal{H}$ if defined, all (previously issued) certificates from honest users are always unmarshaled with label `0`. Due to these facts, we also have that the resulting shadows are always injective for $name$ and $certval$.

For messages in $enctable$, we should notice that if $(msg, \ulcorner M \urcorner^{\mathsf{D}_n,\mathcal{H}}, k) \in enctable$ then, there is an i such that $\mathsf{D}_n.wire(i) = (M, k, id_b\_id_a\_msg, b, true)$ for $b = 0$ or $b = 1$. Now, if $k \in keycache_a$, the message is discarded and we are in Case 2; if $k \notin keycache_a$, then by definition of D_n, $b = 0$, which implies that M/i is defined in S_n, hence there exists $S_n \xrightarrow{(i)} S_n^\bullet$ and, defining $\mathsf{D}_n^\bullet$ as in Lemma C.5, we have that $\overline{\mathsf{N}}(S_n, \mathsf{D}_n) \overset{(inp)}{\rightsquigarrow} \overline{\mathsf{N}}(S_n^\bullet, \mathsf{D}_n^\bullet)$.

As for reduction we may as well apply Lemma C.4 since $\mathsf{D}^{\bullet}_n.name$ and $\mathsf{D}^{\bullet}_n.certval$ are injective by construction and the internal behaviour of $\overline{\mathsf{M}}$ and $\overline{\mathsf{N}}$ is the same, (there is just an injective renaming on indices and labels for certificates) so there exist S^*_n and D^*_n such that $\overline{\mathsf{N}}(S^{\bullet}_n, \mathsf{D}^{\bullet}_n) \rightsquigarrow^* \overline{\mathsf{N}}(S^*_n, \mathsf{D}^*_n)$.

Consider now that $\overline{\mathsf{N}}(S^*_n, \mathsf{D}^*_n) \overset{s}{\rightsquigarrow} \mathsf{N}$. Regarding marshaling, marshaling is only performed when sending messages to the adversary, we have the same as in Lemma C.5 except that, as $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ signs messages for all principals using $[\![\cdot]\!]$ instead of $[\![\cdot]\!]_a$, we should now define $\mathsf{D}'.certval(v_1\{v_2\}_{\mathsf{o}}) = s'$ for each new pair $(v_1\{v_2\}_{\mathsf{o}}, s')$ in $signed$ and not just for certificates form a.

For sending, let S and D as in Lemma C.5 except that we extend D such that for each $M_{i_j} = a{:}u_j\langle V_j\rangle$ to honest principal u_j, $\mathsf{D}.wire(i_j) = (M_{i_j}, k_j, s_{i_j}, 0, \textit{true})$, where k_j is the authentication key produced by the modified $\mathtt{send}_b$ algorithm of Definition C.5 on input M_{i_j}. We have then that $\mathsf{N} = \overline{\mathsf{N}}(S, \mathsf{D})$.

From this, we can show that there exist S with shadow D, and normal transitions $S_n \xrightarrow{\alpha\widetilde{\beta}} S$ such that $\mathsf{N} = \overline{\mathsf{N}}(S, \mathsf{D})$. By construction $\mathsf{D}.name$ and $\mathsf{D}.certval$ are injective. □

Next we show that the probability of failure of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ and $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ are negligible functions of the security parameter η. We do this by reducing the problem of failing to the problem of breaking CMA-security (Definition A.6).

Lemma C.8. *Let S° be a safe initial system with initial shadow D° and $\overline{\mathsf{A}}$ a PPT algorithm. We have $\Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)]$ fails$] \leq \operatorname{neg}(\eta)$.*

Proof. We first recall that, for any CMA-secure signature scheme, finite index set $\mathcal{H}$, and PPT adversaries $\mathsf{Adv}_{\mathsf{CMA}}$, we have:

$$\begin{aligned}\Pr[\quad & (s_a, v_a)_{a\in\mathcal{H}} \longleftarrow \mathcal{G}(1^\eta);\\ & (m, sig) \longleftarrow \mathsf{Adv}_{\mathsf{CMA}}{}^{(\mathcal{S}_a(\cdot))_{a\in\mathcal{H}}}(1^\eta, (v_a)_{a\in\mathcal{H}}) :\\ & \mathcal{V}(v_a, m, sig) = 1 \text{ for some } a \in \mathcal{H} \text{ and } m \notin \mathit{Queries}_a \quad] \leq \operatorname{neg}(\eta)\end{aligned}$$

where each of the oracles $\mathcal{S}_a(x)$ returns $\mathcal{S}(s_a, x)$ and adds x to the set $\mathit{Queries}_a$.

Let S° be a safe initial system with initial shadow D°, and $\overline{\mathsf{A}}$ a PPT algorithm, such that the probability that $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)]$ fails is not a negligible function of η. We define a CMA adversary $\mathsf{Adv}_{\mathsf{CMA}}{}^{(\mathcal{F}_a(\cdot))_{a\in\mathcal{H}}}$ as $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)]$ with the following changes:

- for each $a \in \mathcal{H}$, instead of generating (s_a, v_a), use parameter v_a and set $s_a = 0$;
- for each $a \in \mathcal{H}$, instead of signing with s_a, call the oracle $\mathcal{F}_a(\cdot)$;
- if the failure flag is set, stop and outputs the message that was given as input to $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ just before the failure.

Game 1:

1. $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s$.

Game 2:

1. $(s_a, v_a)_{a\in\mathcal{H}} \longleftarrow \mathcal{G}(1^\eta)$;
2. $(m, sig) \longleftarrow \mathsf{Adv}_{\mathsf{CMA}}{}^{(\mathcal{S}_a(\cdot))_{a\in\mathcal{H}}}(1^\eta, (v_a)_{a\in\mathcal{H}})$.

It is immediate that Game $2 =_d \overline{\mathsf{A}}[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)]$, where $=_d$ denotes the equality of probability distributions, until the run fails. For that just suppose that the keys generated in Step 1 of Definition 3.1 are generated in the same order as in Game 2. We have that the verification keys for each principal in the two games are the same. As for signatures, $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ performs the signatures by itself, while in Game 2 a signing oracle is used, but the outcome of both is the same as $\mathcal{S}(s_a, \cdot) = \mathcal{S}_a(\cdot) = \mathcal{F}_a(\cdot)$.

Let us now analyse the behaviour of $\mathsf{Adv}_{\mathsf{CMA}}$ until the run fails. By Lemma C.7, there exist S, D, φ such that $S^\circ \xrightarrow{\varphi} S$ is a normal transition, D is a shadow for S, and

$$\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \overline{\mathsf{N}}(S, \mathsf{D}) \overset{(inp)}{\rightsquigarrow} \rightsquigarrow^* \overset{s}{\rightsquigarrow} \mathsf{N}$$

such that at the end, the failure flag is set in N, but not in $\overline{\mathsf{N}}(S, \mathsf{D})$.

Suppose that the failure was set upon an input of the form $id_b\_id_a\_msg$. We will start by showing that this never occurs if the message was indeed sent by b, and then that in the case that e is pretending to be b this failure only occurs if e is able to forge a signature.

1. if the message was indeed sent by b then $enctable$ associates msg with $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}, k$ where $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}$ and k are respectively the message passed to the interpreter and the authentication key used. By definition of $enctable$ we have that

$$\textit{there exists } i \textit{ such that } \mathsf{D}.wire(i) = (M, k, id_b\_id_a\_msg, \_, true). \tag{C.2}$$

   On the other hand, if the message was accepted by a, by definition of $keycache$, we have that

$$\textit{there is no } j \textit{ such that } \mathsf{D}.wire(j) = (M', k, \_, 1, \_), \tag{C.3}$$

   where M' is a message to a. By (C.2) and (C.3) we have that for the i above

$$\mathsf{D}.wire(i) = (M, k, id_b\_id_a\_msg, 0, true). \tag{C.4}$$

   From (C.4), and definition of $wire$ we have that $i \in I$, hence M/i is defined in S. For this, a failure can never occur whenever the message was sent by principal b.

2. if the message was not sent by b (i.e., $\overline{\mathsf{A}}$ is trying to create or modify a message pretending to be b) then there is no association between msg and $\ulcorner M \urcorner^{\mathsf{D},\mathcal{H}}, k$ in $enctable$. By definition of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$, this message is accepted only if it is accepted by $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$, that is, it has to pass the `receive` protocol, and in particular $\mathcal{V}(v_b, k\_id_a, s_{sig}) = 1$ for some authentication key k. (condition 2 of Definition 3.23). Notice that a signature of a tuple of the form k_id_a by b is only performed whenever b is sending a message to a. But, by definition of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$, such keys are never signed by b, hence $\mathsf{Adv}_{\mathsf{CMA}}$ was able to forge a signature. From this, the probability that $\mathsf{Adv}_{\mathsf{CMA}}$ outputs a signature from b for a message that was never asked for signing to the signing oracle $\mathcal{F}_b(\cdot)$ is a non-negligible function of η, breaking the CMA security assumption of the signing scheme.

Suppose that the failure was set upon an input of a message $id_e\_id_a\_msg$. By definition of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$, these messages are never in $enctable$, hence $id_e\_id_a\_msg$ was received and unmarshaled as in $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$. In particular, all the certificates form honest principals were verified during the unmarshaling protocol. If the failure flag was set, this implies that the input transition (M) is not enabled, hence some of the certificates in M from honest users are not in $\mathcal{M}(\Phi)$, or some name generated by the adversary was marshaled to a private name. The latter is not possible in $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ as private names are never marshaled, hence no bitstrings sent by the adversary will ever be unmarshaled to identifiers of private names. Hence, if the run fails, some of the certificates in M from honest users are not in $\mathcal{M}(\Phi)$. This implies that these certificates were never sent to the adversary and, by definition of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$, they were never signed, hence the probability that $\mathsf{Adv}_{\mathsf{CMA}}$ outputs a certificate signed by b for which a signature was never performed is a non-negligible function of η, breaking the CMA security assumption of the signing scheme.

Another possibility of failure is by condition 2 of Definition C.1. It is easy to see that the probability of failure due to this is a negligible function of η. For that, notice that by safety of S°, we have that the output of S° has to be polynomial in the size of the input labels, and hence, the number of generated names, n_{names}, is bound above by that polynomial. A clash on these names only occur with negligible probability, more concretely, with probability at most $(n_{\mathit{names}})^2 \times 2^{-\eta}$.

We have shown that if a run of $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ fails with non-negligible probability, then this could not have happened due to a clash of generated names (as this only happen with negligible probability), hence there is an adversary $\mathsf{Adv}_{\mathsf{CMA}}$ that is able to break CMA-security assumption. □

Lemma C.9. *Let S° be a safe initial system with initial shadow D° and $\overline{\mathsf{A}}$ a PPT algorithm. We have* $\Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)]$ *fails*$] \leq \operatorname{neg}(\eta)$.

Proof Sketch. The proof is similar to the proof of Lemma C.8. The only difference lies on the fact that $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ generates more names, hence more failure cases may occur.

Failure upon an input of the form $id_b\_id_a\_msg$ is equal to the case for $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$. The probability of failing upon an input of the form $id_e\_id_a\_msg$, is the probability of producing a fake certificate (equal to the case for $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$) plus the probability of capturing a private name. The probability of the adversary capturing a private name is bound by $(n'_{\mathit{names}})^2 \times 2^{-\eta}$, where n'_{names} is the total number of names generated by $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ (now includes also the names generated to honest principals) that has to be bound by a polynomial since S° is safe. Hence this case fails with negligible probability.

The probability of failing due to condition 2 of Definition C.1 remains negligible, and is at most $(n'_{\mathit{names}})^2 \times 2^{-\eta}$. □

C.4 Reduction to Cryptographic Primitives

In this section, we let S° range over safe initial system with initial shadow D°.

Lemma C.10. $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$.

Proof. Relying on Definition C.4 we can establish the following invariant between the state of $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$ and $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$:

- there is an injective renaming between the name identifiers in $\overline{\mathsf{M}}_a(S^\circ, \mathsf{D}^\circ)$ and the name identifiers in $\mathsf{N}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$; $\mathsf{N}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$ uses the same identifier across machines while $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$ uses a different identifier for each machine.

- for every certificate $\mathtt{prin}(id_a)\{v_2\}_0$ in $\mathsf{N}_{\mathcal{H},a}(S^\circ, \mathsf{D}^\circ)$ there is a certificate $\mathtt{prin}(id_a)\{v_2\}_0$ in $\mathsf{M}_a(S^\circ, \mathsf{D}^\circ)$;

- for every certificate $\mathtt{prin}(id_a)\{v_2\}_0$ there is a signature value s such that for all $b \neq a$, if $\mathtt{prin}(id_a)\{v_2\}_0$ is in $\mathsf{N}_{\mathcal{H},b}(S^\circ, \mathsf{D}^\circ)$ then $\mathtt{prin}(id_a)\{v_2\}_s$ is in $\mathsf{M}_b(S^\circ, \mathsf{D}^\circ)$.

This invariant and Lemma C.1 guarantee that $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$ and $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$ have the same observable behaviour as the internal processes of each machine are the same up to renaming; renaming does not affect the internal computation, and Lemma C.1 ensures that marshaling a bitstring that was previously unmarshaled yields the same result regardless the use of $parse_a(\cdot)$ and $[\![\cdot]\!]_a$ or $parse(\cdot)$ and $[\![\cdot]\!]$. Condition $[\![\ulcorner V \urcorner^{\mathsf{D},a}]\!]_a = [\![\ulcorner V \urcorner^{\mathsf{D},\mathcal{H}}]\!]$ ensures that the initialisation is consistent with this procedure as $p_a = \ulcorner P_a \urcorner^{\mathsf{D},a}$ in $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$ and $p_a = \ulcorner P_a \urcorner^{\mathsf{D},\mathcal{H}}$ in $\mathsf{N}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$. □

Lemma C.11. $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^{\widetilde{0}}(S^\circ, \mathsf{D}^\circ)$.

Proof. Relying on Definition C.5 with $n_a = 0$ for every $a \in \mathcal{H}$, it is immediate that the two machines are equivalent: $j_a = 0$, so the test $j_a < n_a$ always fails, *enctable* remains empty, hence there are no modifications in the protocol. □

Our next lemma deals with the inductive case, by reduction to CCA-2 security.

Lemma C.12. *For some $a \in \mathcal{H}$, let $\widetilde{n} + 1$ abbreviate $\widetilde{n}$ with $n_a + 1$ instead of n_a.*

We have $\Pr[\, \overline{\mathsf{A}}[\overline{\mathsf{N}}^{\widetilde{n}}(S^\circ, \mathsf{D}^\circ)] \longrightarrow 1\,] - \Pr[\, \overline{\mathsf{A}}[\overline{\mathsf{N}}^{\widetilde{n}+1}(S^\circ, \mathsf{D}^\circ)] \longrightarrow 1\,] \leq \operatorname{neg}(\eta)$, *where* $\operatorname{neg}(\cdot)$ *is a function that only depends upon the encryption scheme.*

Proof. Let $\overline{\mathsf{M}}_0 = \overline{\mathsf{N}}^{\widetilde{n}}(S^\circ, \mathsf{D}^\circ)$ and $\overline{\mathsf{M}}_1 = \overline{\mathsf{N}}^{\widetilde{n}+1}(S^\circ, \mathsf{D}^\circ)$. Let $\overline{\mathsf{A}}$ be a PPT adversary. We first rearrange our target property to match the structure of CCA-2 games, and remark that

$$\Pr\left[\begin{array}{l} b \longleftarrow \{0,1\}\,; \\ \overline{\mathsf{A}}[\overline{\mathsf{M}}_b] \longrightarrow s; \\ \text{if } s = 1 \text{ then } g = 1 \text{ else } g = 0 : \\ b = g \end{array}\right] \leq \tfrac{1}{2} + \operatorname{neg}(\eta) \qquad \text{(C.5)}$$

implies that $\Pr[\, \overline{\mathsf{A}}[\overline{\mathsf{M}}_0] \longrightarrow 1\,] - \Pr[\, \overline{\mathsf{A}}[\overline{\mathsf{M}}_1] \longrightarrow 1\,] \leq \operatorname{neg}(\eta)$, from which we can conclude that $\overline{\mathsf{M}}_0 \approx \overline{\mathsf{M}}_1$ by definition of indistinguishability (Definition C.2).

Assume that there exists $\overline{\mathsf{A}}$ that breaks Property (C.5). We then use $\overline{\mathsf{A}}$ and $\overline{\mathsf{N}}^{\tilde{n}}(S^\circ, \mathsf{D}^\circ)$ to build an adversary that breaks CCA-2 security (Definition A.4), recalled below:

$$\Pr\left[\begin{array}{l} b \longleftarrow \{0,1\}\,; \\ (e,d) \longleftarrow \mathcal{K}(1^\eta); \\ m_0, m_1 \longleftarrow {\mathsf{Adv}_{\mathsf{CCA2}}}^{\mathcal{D}_1(\cdot)}(1^\eta, e); \\ c^* \longleftarrow \mathcal{E}(e, m_b); \\ g \longleftarrow {\mathsf{Adv}_{\mathsf{CCA2}}}^{\mathcal{D}_2(\cdot)}(1^\eta, e, c^*) : \\ b = g \end{array}\right] \leq \tfrac{1}{2} + \mathrm{neg}\,(\eta) \tag{C.6}$$

More precisely, we define our CCA2 adversary so that runs of the two games displayed below with the same boolean value for b and the same random inputs always yield the same outcome $b = g$.

Game 1:

1. $\overline{\mathsf{A}}[\overline{\mathsf{M}}_b] \longrightarrow s$;
2. if $s = 1$ then $g = 1$ else $g = 0$.

Game 2:

1. $(e,d) \longleftarrow \mathcal{K}(1^\eta)$;
2. $m_0, m_1 \longleftarrow {\mathsf{Adv}_{\mathsf{CCA2}}}^{\mathcal{D}_1(\cdot)}(1^\eta, e)$;
3. $c^* \longleftarrow \mathcal{E}(e, m_b)$;
4. $g \longleftarrow {\mathsf{Adv}_{\mathsf{CCA2}}}^{\mathcal{D}_2(\cdot)}(1^\eta, e, c^*)$.

We let ${\mathsf{Adv}_{\mathsf{CCA2}}}^{\mathcal{F}(\cdot)}$ be $\overline{\mathsf{A}}[\overline{\mathsf{N}}^{\tilde{n}}(S^\circ, \mathsf{D}^\circ)]$ with the following changes:

1. In Step 1 of Definition 3.1, instead of generating (e_a, d_a) for a, let $e_a = e$ and $d_a = 0$. (e is the key generated at Step 1 of Game 2).
2. Instead of decrypting with d_a, use the oracle $\mathcal{F}(\cdot)$, that is $\mathcal{D}_1(\cdot)$ and $\mathcal{D}_2(\cdot)$, respectively before and after the challenge ciphertext c^*.
3. Change the $\mathtt{send}_a$ protocol (Definition 3.22) for the $n_a + 1$-th message to a as follows: on input m' to the $\mathtt{send}_a$ protocol, instead of encrypting message m at Step 3,
   (a) let $m_0 = m$ and $m_1 = 0^{\lceil m \rceil}$ (Step 2 of Game 2);
   (b) pass m_0 and m_1 to the encryption oracle and continue with $c^* = \mathcal{E}(e, m_b)$ (Step 3 of Game 2);
   (c) record $(c^*, \mathit{parse}(m'), k)$ in $\mathit{enctable}$; and
   (d) resume the $\mathtt{send}_a$ protocol at Step 4.
4. If $\overline{\mathsf{A}}[\overline{\mathsf{N}}^{\tilde{n}}(S^\circ, \mathsf{D}^\circ)] \longrightarrow 1$, output 1, else output 0 (Step 4 of Game 2).

Let $G_1(b)$ and $G_2(b)$ denote respectively Game 1 and Game 2 when the randomly chosen bit is b. We show that the outcome of $G_1(b) = G_2(b)$, for $b \in \{0,1\}$. This is done by induction on the number of low-level steps, showing that the following state invariants are preserved. If $b = 0$:

- either $j_a \leq n_a$ and $G_1(0)$ and $G_2(0)$ have the same state, except for the value of d_a (set to 0 and never used in the second game).
- or $j_a = n_a + 1$ and $G_1(0)$ and $G_2(0)$ have the same state, except for the value of d_a and an extra entry in $enctable$ in the second game, of the form $(c^*, parse(m'), k)$, where m', c^* and k are respectively the input, the ciphertext obtained at Step 3b, and the authentication key, of the modified sending algorithm of $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$.
- Also, a call $\mathcal{F}(c^*)$ is never performed.

If $b = 1$ the following invariant is preserved:

- $G_1(1)$ and $G_2(1)$ have the same state, except for the value of d_a (set to 0 and never used in the second game).
- In addition, a call $\mathcal{F}(c^*)$ is never performed.

Before proving that these invariants are preserved, let us remark that in spite of principal a using different decryption oracles, (in Game 1, compliant principal a performs decryptions using $\mathcal{D}(d_a, \cdot)$, while in Game 2 decryptions are performed by calling decryption oracle $\mathcal{F}(\cdot)$, change 2 in $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$), the outcome of these decryptions is the same as long as $d = d_a$ and no call $\mathcal{F}(c^*)$ is made. (This assures, that d_a is never used in the second game.) This is true since, by definition, $\mathcal{F}(c) = \mathcal{D}(d_a, c)$ for all $c \neq c^*$, and $\perp$ otherwise, hence all messages $c \neq c^*$ will be decrypted to the same values regardless the use of $\mathcal{F}(\cdot)$ or $\mathcal{D}(d_a, \cdot)$.

Let us start by considering that $b = 0$ and let us analyse the behaviour of the two games on the same random inputs. Suppose that we have a run of length n. We will analyse the cases where the two games have apparent different behaviours.

Base case $n = 0$: In this case, only initialisation is performed and $j_a = 0 \leq n_a$. We have to show that the invariant is preserved.

The initialisation of the two games is different. In Game 1, initialisation is performed as described in Step 1 of Definition 3.1, while in Game 2 we replace the generation of (e_a, d_a) by $(e, 0)$ where $(e, d) \longleftarrow \mathcal{K}(1^\eta)$ is the pair generated in Step 1 of Game 2 (change 1 in $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$). Since the order of generation of the cryptographic material in Step 1 of Definition 3.1 is irrelevant, we may assume, without loss of generality, that the pair (e_a, d_a) is the first one to be generated, hence $(e_a, d_a) = (e, d)$. (This establishes the first part of the invariant, the state of the two games is the same except for d_a that is set to 0 in the second game.)

Also, since no more steps were performed, d_a was never used in the second game. Thus, the base case establishes the invariant.

Inductive case: Let us now suppose that after n low-level steps, the state of the two games satisfy the invariant. Consider first the case $j_a \leq n_a$:

1. an input is provided to principal a:

(a) an input $id_u\_id_v\_msg$ is provided to principal a and msg is associated with some m, k in $enctable$. Since the state of $G_1(0)$ and $G_2(0)$ is the same, so is $enctable$, hence the two games behave the same way. In this case the invariant is preserved.

(b) an input $id_u\_id_v\_msg$ is provided to principal a but msg is not associated with any m, k in $enctable$. In this case both $G_1(0)$ and $G_2(0)$ behave as $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$ (Definition C.5), except that $G_2(0)$ will not use $\mathcal{D}(d_a, \cdot)$ but instead $\mathcal{F}(\cdot)$. As remarked above ($d = d_a$, and $msg \neq c^*$ since $j_a \leq n_a$) the decryptions of these messages will lead to the same value, hence $G_1(0)$ and $G_2(0)$ behave the same way. The invariant is also preserved in this case.

2. an input is provided to compliant principal c:

(a) an input $id_u\_id_v\_msg$ is provided to compliant principal c and msg is associated with some m, k in $enctable$. This case is equal to Case 1a.

(b) an input $id_u\_id_v\_msg$ is provided to compliant principal c but msg is not associated with any m, k in $enctable$. In this case both $G_1(0)$ and $G_2(0)$ behave as $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$ (Definition C.5). The invariant is also preserved in this case.

3. an internal reduction is performed by some principal c. In this case the invariant is trivially preserved as there is no difference from $G_1(0)$ to $G_2(0)$.

4. principal a performs an output. Since a never sends messages to itself, after this output $j_a \leq n_a$ and the invariant is preserved.

5. compliant principal c performs an output: in this case there is no difference between $G_1(0)$ and $G_2(0)$ except when sending messages to a. Let us analyse the sending procedure $\mathtt{send}_a$ of c:

(a) after sending all messages, $j_a \leq n_a$. In this case the $n_a + 1$-th message to a was not sent hence $G_1(0)$ and $G_2(0)$ behave as in Definition C.5, and the invariant is preserved.

(b) after sending all messages, $j_a = n_a + 1$. In this case, the $n_a + 1$-th message to a was sent. For this particular message $G_1(0)$ behaves as $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$, while $G_2(0)$ behaves as provided in change 3 of $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$, that is, for $b = 0$ it gets $c^* = \mathcal{E}(e_a, m)$ and records $(c^*, parse(m'), k)$ in $enctable$, where m' and k are respectively the input and the authentication key, of the modified sending algorithm of $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$. After sending this message we have that $j_a = n_a + 1$ and the state of $G_2(0)$ is equal to the state of $G_1(0)$ with this extra entry in $enctable$. This satisfies the invariant.

Let us now consider the case $j_a = n_a + 1$:

1. an input is provided to principal a:

(a) an input $id_u\_id_v\_msg$ is provided to principal a and msg is associated with some m, k in both $enctable$ of $G_1(0)$ and $G_2(0)$. This case is similar to Case 1a of $j_a \leq n_a$.

(b) an input $id_u\_id_v\_msg$ is provided to principal a and msg is associated with some m, k in $enctable$ of $G_2(0)$ but not of $G_1(0)$. The only entry that satisfies such conditions is $(c^*, parse(m'), k)$. In this case $G_2(0)$ accepts the message if and only if u is the sender of m', $v = a$, $k \notin keycache_a$, and adds k to $keycache_a$ returning $parse(m')$ to the interpreter.

As for $G_1(0)$, it behaves as $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$, that is, receives the message, decrypts it, performs the checks of Definition 3.22, adds k to $keycache_a$, and returns m', that was the input to the sending algorithm. It then passes $parse(m')$ to the interpreter, as $G_2(0)$, hence the behaviour of the two games is the same, and the invariant is preserved.

(c) an input $id_u\_id_v\_msg$ is provided to principal a and msg is not associated with any m, k in either $enctable$ of $G_1(0)$ or $G_2(0)$. This case is similar to Case 1b of $j_a \leq n_a$ noticing that $msg \neq c^*$ as this would imply that the message is in $enctable$ of $G_2(0)$.

2. an input is provided to compliant principal c:

   (a) an input $id_u\_id_v\_msg$ is provided to compliant principal c and msg is associated with some m, k in $enctable$. This case is equal to Case 2a of $j_a \leq n_a$.

   (b) an input $id_u\_id_v\_msg$ is provided to compliant principal c but msg is not associated with any m, k in $enctable$. This case is equal to Case 2b of $j_a \leq n_a$.

   It never occurs that an input $id_u\_id_v\_msg$ is provided to principal c and msg is associated with some m, k in $enctable$ of $G_2(0)$ but not of $G_1(0)$. This only occur in messages to a.

3. an internal reduction is performed by some principal c. In this case the invariant is trivially preserved as there is no difference from $G_1(0)$ to $G_2(0)$.

4. principal a performs an output. Since a never sends messages to itself, after this output $j_a \leq n_a$ and the invariant is preserved.

5. compliant principal c performs an output: in this case there is no difference between $G_1(0)$ and $G_2(0)$ except when sending messages to a. Let us analyse the sending procedure $\mathtt{send}_a$ of c:

   (a) as $j_a = n_a + 1$, both $G_1(0)$ and $G_2(0)$ behave as $\overline{\mathsf{N}}_{\mathcal{H}}(S^\circ, \mathsf{D}^\circ)$, and the state remains the same, with an extra entry in $enctable$ in the second game, of the form $(c^*, parse(m'), k)$, hence the invariant is preserved.

Let us now consider the case $b = 1$ and let us analyse the behaviour of the two games on the same random inputs. Suppose that we have a run of length n.

Base case $n = 0$: This case is equal to the one proven above for $b = 0$.

Inductive case: Let us now suppose that after n low-level steps, the state of the two games satisfy the invariant.

1. an input is provided to principal a: this case is equal to Case 1 of $j_a \leq n_a$ for $b = 0$.

2. an input is provided to principal c: this case is equal to Case 2 of $j_a \leq n_a$ for $b = 0$.

3. an internal reduction is performed by some principal c. In this case the invariant is trivially preserved as there is no difference from $G_1(1)$ to $G_2(1)$.

4. principal a performs an output. Since a never sends messages to itself, after this output $j_a \leq n_a$ and the invariant is preserved.

5. compliant principal c performs an output: in this case there is no difference between $G_1(1)$ and $G_2(1)$ except when sending messages to a. The sending procedures only differ in the case $j_a = n_a$, that is whenever sending the $n_a + 1$-th message to a. Let us analyse the sending procedure $\mathtt{send}_a$ of c:

   (a) after sending all messages, $j_a \leq n_a$. In this case the $n_a + 1$-th message to a was not sent hence $G_1(0)$ and $G_2(0)$ behave as in Definition C.5, and the invariant is preserved.

   (b) after sending all messages, $j_a = n_a + 1$. In this case a message the $n_a + 1$-th message to a was sent to a, that is, a message was sent with $j_a = n_a$. In this case, $G_1(1)$ ($\overline{\mathsf{N}}^{\tilde{n}+1}(S^\circ, \mathsf{D}^\circ)$) behaves as in Definition C.5, that is it marshals the content of m to m', encrypts 0's and records $(\mathcal{E}(e_a, 0^{\lceil m'' \rceil}), m, k)$ in $enctable$, where m'' is the message obtained at Step 3 of the $\mathtt{send}_a$ protocol on input m'. $G_2(1)$ behaves as provided in change 3 of $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$, that is, for $b = 1$, on input m' to the $\mathtt{send}_a$ protocol, it gets $c^* = \mathcal{E}(e_a, 0^{\lceil m'' \rceil})$ and records $(c^*, parse(m'), k)$ in $enctable$, where m' and k are respectively the input and the authentication key, of the modified sending algorithm of $\mathsf{Adv_{CCA2}}^{\mathcal{F}(\cdot)}$. Since $m = parse(m')$ the invariant is preserved.

□

We now relate machines $\overline{\mathsf{N}}^{\tilde{\omega}}$ and $\overline{\mathsf{N}}^{\star}$ that differ only in the marshalling of certificates for messages exchanged between compliant principals. $\overline{\mathsf{N}}^{\star}$ does not marshal certificates when sending messages to honest principals, while $\overline{\mathsf{N}}^{\tilde{\omega}}$ generates the certificates but does not send them.

Lemma C.13. $\overline{\mathsf{N}}^{\tilde{\omega}}(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^{\star}(S^\circ, \mathsf{D}^\circ)$.

Proof. Let $\overline{\mathsf{A}}$ be a PPT adversary. Consider machine $\overline{\mathsf{N}}^{\sharp}(S^\circ, \mathsf{D}^\circ)$ derived from $\overline{\mathsf{N}}^{\tilde{\omega}}(S^\circ, \mathsf{D}^\circ)$ as follows (in the following, we denote by $signed^\sharp$ the table $signed$ of machine $\overline{\mathsf{N}}^{\sharp}(S^\circ, \mathsf{D}^\circ)$):

- we extend the state of $\overline{\mathsf{N}}^{\sharp}(S^\circ, \mathsf{D}^\circ)$ with a table *hidden-cert* that associates internal representations of certificates to signature values;

- whenever marshaling a message to $b \in \mathcal{H}$, for each certificate of the form $\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}}$ such that both $signed^\sharp(\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}})$ and *hidden-cert*$(\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}})$ are undefined, add $(\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}}, \mathcal{S}(s_a, [\![v_2]\!]))$ to *hidden-cert* instead of adding it to $signed^\sharp$;
- whenever marshaling a message to $e \notin \mathcal{H}$, if it contains a certificate $\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}}$ for some $(\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}}, s) \in$ *hidden-cert*, marshal as $\mathtt{cert}\_[\![\mathtt{prin}(id_a)]\!]\_[\![v_2]\!]\_s$ and add $(\mathtt{prin}(id_a)\{v_2\}_{\mathtt{0}}, s)$ to $signed^\sharp$.

It is easy to see that $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ (in fact we even have equality of distributions). Machine $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$ generates the certificates upfront but it only uses them whenever they are needed in $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$, that is, whenever they are part of a message to the adversary. For that, the two machines have the same distribution. Moreover, since both machines define exactly the same certificates (the ones generated *a priori* by $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$ do not interfere with the computation unless they are added to $signed^\sharp_a$) the probability of failure of both machines is also the same.

It remains to show that $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$. We claim that for any run with length n

$$\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_n}{\rightsquigarrow} \mathsf{N}^\sharp$$

if the failure flag is not set in $\mathsf{N}^\sharp$, then

1. there exist S with shadow $\mathsf{D}^\sharp$ and normal transitions $S^\circ \xrightarrow{\varphi} S$, such that $\mathsf{N}^\sharp = \overline{\mathsf{N}}^\sharp(S, \mathsf{D}^\sharp)$, and
2. $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_n}{\rightsquigarrow} \overline{\mathsf{N}}^{\widetilde{\omega}}(S, \mathsf{D})$, the failure flag is not set in $\overline{\mathsf{N}}^{\widetilde{\omega}}(S, \mathsf{D})$, and D is obtained from $\mathsf{D}^\sharp$ by adding $\mathsf{D}.certval(cert) = s$ for all certificates $cert$ such that $(cert, s) \in$ *hidden-cert*.

It follows from our claim that no adversary can distinguish the two machines when the run of $\mathsf{N}^\sharp(S^\circ, \mathsf{D}^\circ)$ does not fail; the outputs for successful runs are the same in both cases. Hence the only way to distinguish $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^\circ, \mathsf{D}^\circ)$ from $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$ is if $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$ fails. But, $\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ and by Lemma C.9, $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ fails with negligible probability. Hence $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$ as needed.

Let us now prove the claim above.

Base case $n = 0$: We have $\mathsf{N}^\sharp = \overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ)$. We use $S = S^\circ, \mathsf{D}^\sharp = \mathsf{D}^\circ, \varphi = \varepsilon$, and $\mathsf{D} = \mathsf{D}^\circ$. Since *hidden-cert* is empty, D° satisfies the condition in Case 2.

Inductive case: Suppose that

$$\overline{\mathsf{N}}^\sharp(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_1}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^*\overset{s_n}{\rightsquigarrow} \mathsf{N}^\sharp_n \overset{(inp_{n+1})}{\rightsquigarrow}\rightsquigarrow^*\overset{s_{n+1}}{\rightsquigarrow} \mathsf{N}^\sharp \tag{C.7}$$

and the failure flag is not set in $\mathsf{N}^\sharp$. By induction hypothesis for n,

(i) there exist S_n with shadow $\mathsf{D}^\sharp_n$ and normal transitions $S^\circ \xrightarrow{\varphi} S_n$, such that $\mathsf{N}^\sharp_n = \overline{\mathsf{N}}^\sharp(S_n, \mathsf{D}^\sharp_n)$, and

(ii) $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \ldots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$, the failure flag is not set in $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$, and D_n is obtained from $\mathsf{D}^\sharp_n$ by adding $\mathsf{D}_n.certval(cert) = s$ for all certificates $cert$ such that $(cert, s) \in$ ***hidden-cert***.

The proof that there exist S with shadow $\mathsf{D}^\sharp$ and normal transitions $S_n \xrightarrow{\alpha\widetilde{\beta}} S$ such that $\mathsf{N}^\sharp = \overline{\mathsf{N}}^\sharp(S, \mathsf{D}^\sharp)$ whenever the flag is not set in $\mathsf{N}^\sharp$, is similar to the proof of Lemma C.7 so we omit it here.

To show the second part of Property (C.7) we just need to show that on input (inp_{n+1}) (that does not set the failure flag in $\mathsf{N}^\sharp$ by hypothesis) $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ will unmarshal the input to the same value; we need also to show the conditions on the output. Let us start analysing the unmarshaling procedure.

One should recall that $parse(\cdot)$ is only applied when receiving messages from the adversary or when receiving messages from honest users that are not in $enctable$, i.e., messages that the adversary is trying to input as messages sent by b. Hence, if $(inp_{n+1}) = id_b\_id_a\_msg$ and msg is in $enctable$ associated with some $\ulcorner M \urcorner^{\mathsf{D}^\sharp_n, \mathcal{H}}, k$, then $\mathsf{N}^\sharp_n$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ behave the same way as both skip unmarshal and return $\ulcorner M \urcorner^{\mathsf{D}^\sharp_n, \mathcal{H}}$ to the interpreter. Notice that $\ulcorner M \urcorner^{\mathsf{D}^\sharp_n, \mathcal{H}} = \ulcorner M \urcorner^{\mathsf{D}_n, \mathcal{H}}$ as $\mathsf{D}^\sharp_n$ and D_n only differ in certificates with label $\mathsf{0}$ and these, whenever interpreted using $\ulcorner \cdot \urcorner^{\mathsf{D}, \mathcal{H}}$ will have label $\mathsf{0}$ regardless the value defined for it in D.

Suppose that $\mathsf{N}^\sharp_n$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ unmarshal the message to two different internal representations $m^\sharp = \ulcorner M \urcorner^{\mathsf{D}', \mathcal{H}}$ and $m^{\widetilde{\omega}}$, where D' is an extension of $\mathsf{D}^\sharp_n$. If these two representations are different, it implies that there is at least one certificate from an honest principal b such that

$$signed^{\widetilde{\omega}}(\mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}}) = s \quad \text{and} \quad \mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}} \quad \text{is in } m^{\widetilde{\omega}} \tag{C.8}$$
$$signed^{\sharp}(\mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}}) = \perp \quad \text{and} \quad \mathtt{prin}(id_b)\{v_2\}_{s} \quad \text{is in } m^{\sharp} \tag{C.9}$$

as the only difference between the two machines is the content of their $signed$ tables.

Let us consider now the case where $(inp_{n+1}) = id_c\_id_a\_msg$ but msg is not associated with any $\ulcorner M \urcorner^{\mathsf{D}^\sharp_n, \mathcal{H}}, k$ in $enctable$. Since the failure flag is not set in $\mathsf{N}^\sharp$, M/i is defined in S_n (where $m^\sharp = \ulcorner M \urcorner^{\mathsf{D}', \mathcal{H}}$). Let M' be the certificate contained in M such that $\ulcorner M' \urcorner^{\mathsf{D}', \mathcal{H}} = \mathtt{prin}(id_b)\{v_2\}_s$. By Definition 3.25, M' cannot be of the form $b\{V_2\}_{\mathsf{0}}$ otherwise $\ulcorner M' \urcorner^{\mathsf{D}', \mathcal{H}} = \mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}}$, hence $M' = b\{V_2\}_\ell$ for some $\ell \neq \mathsf{0}$. Since $\mathsf{D}^\sharp_n$ is a shadow for S_n, M/i is defined in S_n, and $b\{V_2\}_\ell$ is a subterm of M with $\ell \neq 0$, we have that $\mathsf{D}^\sharp_n(b\{V_2\}_\ell) \neq \perp$, which implies that $\mathsf{D}_n(b\{V_2\}_\ell) \neq \perp$. By Definition 3.25, $\ulcorner b\{V_2\}_\ell \urcorner^{\mathsf{D}_n, \mathcal{H}} = \mathtt{prin}(id_b)\{v_2\}_s$ for some $s \neq \mathsf{0}$ which contradicts the hypothesis that $\mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}}$ is in $m^{\widetilde{\omega}}$ (C.8). Hence $\mathsf{N}^\sharp$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ cannot unmarshal this input to two different internal representations.

Let us now consider the last case: $(inp_{n+1}) = id_e\_id_a\_msg$. Since the failure flag is not set in $\mathsf{N}^\sharp$, S_n has the input transition (M), hence $\phi(M)_{\{b\}} \subseteq \mathcal{M}(\Phi_n)$, in particular $b\{V_2\}_{\mathsf{0}} \in \Phi_n$ for $b\{V_2\}_{\mathsf{0}}$ such that $\ulcorner b\{V_2\}_\ell \urcorner^{\mathsf{D}', \mathcal{H}} = \mathtt{prin}(id_b)\{v_2\}_s$. But, by definition of $\mathsf{D}^\sharp_n$ being a shadow for

S_n, if $b\{V_2\}_{\mathsf{0}} \in \Phi_n$ then all its names and certificates are defined in $\mathsf{D}^\sharp_n$, hence

$$\begin{array}{rcll} \mathit{signed}^\sharp(\ulcorner b\{V_2\}_{\mathsf{0}}\urcorner^{\mathsf{D}^\sharp_n,\mathcal{H}}) & = & \mathit{signed}^\sharp(\ulcorner b\{V_2\}_{\mathsf{0}}\urcorner^{\mathsf{D}',\mathcal{H}}) & \\ & = & \mathit{signed}^\sharp_a(\mathtt{prin}(id_b)\{v_2\}_{\mathsf{0}}) & \text{by Definition 3.25} \\ & = & \bot & \text{by (C.9)} \\ \mathit{signed}^\sharp(\ulcorner b\{V_2\}_{\mathsf{0}}\urcorner^{\mathsf{D}^\sharp_n,\mathcal{H}}) & = & \mathsf{D}^\sharp_n.\mathit{certval}(b\{V_2\}_{\mathsf{0}}) & \text{by Definition C.4} \\ & \neq & \bot & \text{because } b\{V_2\}_{\mathsf{0}} \in \Phi_n \end{array}$$

We derive a contradiction, hence on input $id_e\_id_a\_msg$, machines $\mathsf{N}^\sharp_n$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ cannot unmarshal the message to two different internal representations.

As for internal reductions, since both machines unmarshal the message to the same internal representation, our deterministic scheduler performs the same reductions in both cases.

As for outputs $\mathsf{N}^\sharp_n$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ are also able to produce the same output: either it is a message to $b \in \mathcal{H}$, and in this case $\mathsf{N}^\sharp_n$ adds a tuple $(\mathit{cert}, \mathit{sig})$ to ***hidden-cert*** while $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ adds it to signed, or it is a message to $e \notin \mathcal{H}$ and in this case $\mathsf{N}^\sharp_n$ uses the same certificate as $\overline{\mathsf{N}}^{\widetilde{\omega}}(S_n, \mathsf{D}_n)$ either because both generate it or because it was fetched from ***hidden-cert***. The condition on D is trivially verified.

Hence the two machines have the same behaviour as the only difference among them does not generate different behaviour. $\square$

Similarly, we relate machines $\overline{\mathsf{N}}^\star$ and $\overline{\mathsf{N}}$ that differ only when marshalling names exchanged between compliant principals. $\overline{\mathsf{N}}$ never generates the names in messages to honest principals while $\overline{\mathsf{N}}^\star$ generates the names but do not send them.

Lemma C.14. $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$.

Proof. Let $\overline{\mathsf{A}}$ be a PPT adversary. Consider machine $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ derived from $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ as follows (in the following, we denote by $\mathit{names}^\dagger$ the table names of machine $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$):

- we extend the state of $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ with a table ***hidden*** that associates internal representations of names to bitstrings;
- whenever marshaling a message to $a \in \mathcal{H}$, for each ind such that both $\mathit{names}^\dagger(\mathit{ind})$ and ***hidden***(ind) are undefined, add $(\mathit{ind}, s \longleftarrow \{0,1\}^*)$ to ***hidden*** instead of adding it $\mathit{names}^\dagger$;
- whenever marshaling a message to $e \notin \mathcal{H}$, if it contains ind for some $(\mathit{ind}, s) \in$ ***hidden***, marshal as $\mathtt{name\_}s$ and add (ind, s) to $\mathit{names}^\dagger$.

It is easy to see that $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$ (in fact, we have $=_d$). Machine $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ generates the names upfront but it only uses them whenever they are needed in $\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$. For that, the two machines have the same distribution. Moreover, since both machines define exactly the same names (the ones generated *a priori* by $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ do not interfere with the computation unless they are added to $\mathit{names}^\dagger$) the probability of failure of both machines is also the same.

It remains to show that $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$. We claim that for any run with length n

$$\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ) \overset{(\mathit{inp}_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(\mathit{inp}_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(\mathit{inp}_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{N}^\dagger$$

if the failure flag is not set in $\mathsf{N}^\dagger$, then there exist S with shadow $\mathsf{D}^\dagger$ and normal transitions $S^\circ \xrightarrow{\varphi} S$, such that $\mathsf{N}^\dagger = \overline{\mathsf{N}}^\dagger(S, \mathsf{D}^\dagger)$, and either:

1. $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_n}{\rightsquigarrow} \overline{\mathsf{N}}^\star(S, \mathsf{D}^\star)$, the failure flag is not set in $\overline{\mathsf{N}}^\star(S, \mathsf{D}^\star)$, and $\mathsf{D}^\star$ is obtained from $\mathsf{D}^\dagger$ by adding $\mathsf{D}^\star.name(n) = s$ for all the names n such that $(\mathsf{D}^\dagger.ni(n), s) \in$ *hidden*; or

2. $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_n}{\rightsquigarrow} \mathsf{N}^\star$, and the failure flag is set in $\mathsf{N}^\star$.

It follows from our claim that no adversary can distinguish the two machines when both runs do not fail; the outputs for successful runs are the same in both cases. Hence the only way to distinguish $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ from $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ is if one fails and the other does not. By Lemma C.9, $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ)$ fails with negligible probability. By Lemma C.8 and $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)$, we have that $\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ also fails with negligible probability. Hence, $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$ and the claim follows.

Let us now prove the claim above.

Base case $n = 0$: We have $\mathsf{N}^\dagger = \overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ)$. We use $S = S^\circ, \mathsf{D}^\dagger = \mathsf{D}^\circ, \varphi = \varepsilon$, and $\mathsf{D}^\star = \mathsf{D}^\circ$. Since *hidden* is empty, D° satisfies the condition in Case 1.

Inductive case: Suppose that

$$\overline{\mathsf{N}}^\dagger(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_1}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_n}{\rightsquigarrow} \mathsf{N}^\dagger_n \overset{(inp_{n+1})}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_{n+1}}{\rightsquigarrow} \mathsf{N}^\dagger$$

and the failure flag is not set in $\mathsf{N}^\dagger$. By induction hypothesis for n, there exist S_n with shadow $\mathsf{D}^\dagger_n$ and normal transitions $S^\circ \xrightarrow{\varphi} S_n$, such that $\mathsf{N}^\dagger_n = \overline{\mathsf{N}}^\dagger(S_n, \mathsf{D}^\dagger_n)$, and either:

(i) $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_n}{\rightsquigarrow} \mathsf{N}^\star_n = \overline{\mathsf{N}}^\star(S_n, \mathsf{D}^\star_n)$, the failure flag is not set in $\overline{\mathsf{N}}^\star(S_n, \mathsf{D}^\star_n)$, and $\mathsf{D}^\star_n$ is obtained from $\mathsf{D}^\dagger_n$ by adding $\mathsf{D}^\star_n.name(n) = s$ for all the names n such that $(\mathsf{D}^\dagger_n.ni(n), s) \in$ *hidden*; or

(ii) $\overline{\mathsf{N}}^\star(S^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_1}{\rightsquigarrow}\overset{(inp_2)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow}\rightsquigarrow^{*}\overset{s_n}{\rightsquigarrow} \mathsf{N}^\star_n$, and the failure flag is set in $\mathsf{N}^\star_n$.

The proof that there exist S with shadow $\mathsf{D}^\dagger$ and normal transitions $S_n \xrightarrow{\alpha\tilde{\beta}} S$ such that $\mathsf{N}^\dagger = \overline{\mathsf{N}}^\dagger(S, \mathsf{D}^\dagger)$ whenever the flag is not set in $\mathsf{N}^\dagger$, is similar to the proof of Lemma C.7 so we omit it here.

To show the second part of Property (C.10) we just need to show that on input (inp_{n+1}), machine $\mathsf{N}^\star_n$ will behave the same way as $\mathsf{N}^\dagger_n$ unless it fails. Since $\mathsf{N}^\star_n$ and $\mathsf{N}^\dagger_n$ only differ in their $names$ table ($\mathsf{N}^\star_n$ has more names in $names^\star$), differences between the two may occur when unmarshaling an input leads to different internal representations, $m^\star$ and $m^\dagger$. The other possible difference is when marshaling a message; clashes occur more often in the case of $\mathsf{N}^\star_n$ but, as this implies that $\mathsf{N}^\star$ has the failure flag set to true, this is included in Case 2.

Let us then analyse what may occur when receiving a message. First, suppose that $inp_{n+1} = id_u\_id_a\_msg$ and msg is associated with $\ulcorner M \urcorner^{\mathsf{D}^\dagger_n,\mathcal{H}}, k$ in $enctable$. In this case, the two machines behave the same way as both skip unmarshaling and return $\ulcorner M \urcorner^{\mathsf{D}^\dagger_n,\mathcal{H}}$ to the interpreter. Notice that $\ulcorner M \urcorner^{\mathsf{D}^\dagger_n,\mathcal{H}} = \ulcorner M \urcorner^{\mathsf{D}^\star_n,\mathcal{H}}$ for messages in $enctable$ since $\mathsf{D}^\dagger_n$ and $\mathsf{D}^\star_n$ only differ in $name(m)$ for names never sent to the adversary, but for messages in $enctable$, names are represented by their internal identifiers that are equal in $\mathsf{D}^\dagger_n$ and $\mathsf{D}^\star_n$.

Suppose now that $inp_{n+1} = id_b\_id_a\_msg$ but msg is not associated with any $\ulcorner M \urcorner^{\mathsf{D}^\dagger_n,\mathcal{H}}, k$ in $enctable$. Since the failure flag is not set in $\mathsf{N}^\dagger$ the message is unmarshaled to some $\ulcorner M^\dagger \urcorner^{\mathsf{D}',\mathcal{H}}$ and $M^\dagger/i$ is defined in S_n. Suppose that the message is received and unmarshaled to two different representations

$$\begin{aligned} m^\dagger &= \ulcorner M^\dagger \urcorner^{\mathsf{D}',\mathcal{H}} && \text{where } \mathsf{D}' \text{ extends } \mathsf{D}^\dagger_n\text{, and} \\ m^\star &= \ulcorner M^\star \urcorner^{\mathsf{D}'',\mathcal{H}} && \text{where } \mathsf{D}'' \text{ extends } \mathsf{D}^\star_n \end{aligned}$$

by $\mathsf{N}^\dagger_n$ and $\mathsf{N}^\star_n$. It is easy to see that receive and unmarshaling succeed in $\mathsf{N}^\dagger_n$ if and only if it succeeds in $\mathsf{N}^\star_n$ (verification of the authentication key is the same and similarly for verification of certificates during unmarshaling). Hence if $m^\dagger \neq m^\star$ there is an s' that during the unmarshaling procedure

$$\begin{aligned} parse(s') &= v_1 && \text{in the case of } \mathsf{N}^\dagger_n\text{, and} \\ parse(s') &= v_2 && \text{in the case of } \mathsf{N}^\star_n \end{aligned} \tag{C.10}$$

for $v_1 \neq v_2$. Besides the tables $names^\dagger$ and $names^\star$, there is no other difference between $\mathsf{N}^\dagger_n$ and $\mathsf{N}^\star_n$, hence the unmarshaling of s' differ if and only if $s' = \texttt{name\_}s$.

We have either $(v_1, s) \in names^\dagger$ or $(v_2, s) \in names^\star$ (we cannot have both undefined otherwise s' would have been unmarshaled to a common v). By construction, $names^\dagger \subseteq names^\star$, hence $(v_2, s) \in names^\star$ and

$$\mathsf{D}^\star.name(n) = s \text{ and } \mathsf{D}^\star.ni(n) = v_2 \text{ for some name } n \tag{C.11}$$

by Definition C.4.We have also that

$$(x, s) \notin names^\dagger \qquad \text{for all } x \tag{C.12}$$

by definition of $\mathsf{N}^\dagger$, $names^\dagger \subseteq names^\star$, hence $(x, s) \in names^\dagger$ if and only if $x = v_2$; but $parse(\texttt{name\_}s) = v_1$ in $\mathsf{N}^\dagger_n$ and $v_1 \neq v_2$ by hypothesis, hence there is no x such that $names^\dagger(x) = s$.

By (C.10) and (C.12) we have that v_1 is a new name identifier (created by the $parse(\cdot)$ procedure) that is not defined in $\mathsf{D}^\dagger.ni$. We have then that there exists a name $n' \neq n$ in $M^\dagger$ with

$$\mathsf{D}'.name(n') = s \text{ and } \mathsf{D}'.ni(n') = v_1.$$

Since D' is an extension of $\mathsf{D}^\dagger_n$ and $\mathsf{D}'.name(n') = s$ we have that either $\mathsf{D}^\dagger_n.name(n') = s$ or $\mathsf{D}^\dagger_n.name(n') = \perp$; if $\mathsf{D}^\dagger_n.name(n') = s$ then by hypothesis $\mathsf{D}^\star_n.name(n') = s$. Using (C.11) and injectivity of $\mathsf{D}^{\star}_n.name$ this implies that $n' = n$ which contradicts the fact of n' being a new name; hence

$$\mathsf{D}^{\dagger}_n.name(n') = \bot \text{ and } \mathsf{D}^{\dagger}_n.ni(n') = \bot \tag{C.13}$$

By definition of shadow, $\mathsf{D}^{\dagger}_n.ni$ is undefined at most for locally restricted names, hence n' is a locally restricted name or it is a name generated by the adversary. Obviously n' cannot be locally restricted since it part of $M^{\dagger}/i$, hence it has to be generated by the adversary. In this case, it is free in S_n and by definition of shadow, $\mathsf{D}^{\dagger}_n.name(n') \neq \bot$ which contradicts (C.13), hence on input $id_b\_id_a\_msg$, machines $\mathsf{N}^{\dagger}_n$ and $\mathsf{N}^{\star}_n$ cannot unmarshal the message to two different internal representations.

As the last case, suppose that $inp_{n+1} = id_e\_id_a\_msg$. Suppose again that the message is received and unmarshaled to two different representations

$$m^{\dagger} = \ulcorner M^{\dagger} \urcorner^{\mathsf{D}',\mathcal{H}} \qquad \text{where } \mathsf{D}' \text{ extends } \mathsf{D}^{\dagger}_n\text{, and}$$
$$m^{\star} = \ulcorner M^{\star} \urcorner^{\mathsf{D}'',\mathcal{H}} \qquad \text{where } \mathsf{D}'' \text{ extends } \mathsf{D}^{\star}_n$$

by $\mathsf{N}^{\dagger}_n$ and $\mathsf{N}^{\star}_n$. Similarly to the previous case we obtain that $(v_2, s) \in names^{\star}$ and there is no x such that $(x, s) \notin names^{\dagger}$. We have then that

$$\begin{aligned}
& (v_2, s) \in names^{\star} \\
\Longrightarrow\quad & \exists n : \mathsf{D}^{\star}_n.name(n) = s \ \wedge\ \mathsf{D}^{\star}_n.ni(n) = v_2 && \text{(C.14)}\\
\Longrightarrow\quad & \exists n : \mathsf{D}^{\star}_n.name(n) = s \ \wedge\ \mathsf{D}^{\dagger}_n.ni(n) = v_2 && \text{(C.15)}\\
\Longrightarrow\quad & \exists n : \mathsf{D}^{\star}_n.name(n) = s \ \wedge\ \mathsf{D}^{\dagger}_n.ni(n) = v_2 \ \wedge \\
& \qquad\qquad \wedge\ (v_2, \mathsf{D}^{\dagger}_n.name(n)) \in names^{\dagger} && \text{(C.16)}\\
\Longrightarrow\quad & \exists n : \mathsf{D}^{\star}_n.name(n) = s \ \wedge\ \mathsf{D}^{\dagger}_n.ni(n) = v_2 \ \wedge\ \mathsf{D}^{\dagger}_n.name(n) \neq s && \text{(C.17)}\\
\Longrightarrow\quad & \exists n : \mathsf{D}^{\dagger}_n.name(n) = \bot \ \wedge\ \mathsf{D}^{\dagger}_n.ni(n) = v_2 \ \wedge \\
& \qquad\qquad \wedge\ (\mathsf{D}^{\dagger}_n.ni(n), s) \in \textit{hidden} && \text{(C.18)}\\
\Longrightarrow\quad & (v_2, s) \in \textit{hidden} \ \wedge \\
& \qquad\qquad \wedge\ \mathsf{D}^{\dagger}_n.name(n) = \bot \text{ for } n \text{ such that } \mathsf{D}^{\dagger}_n.ni(n) = v_2 && \text{(C.19)}
\end{aligned}$$

Step (C.14) follows from Definition C.4 applied to $names^{\star}$. Step (C.15) follows from the definition of $\mathsf{N}^{\dagger}$ machine; it has the same internal representations for names as $\mathsf{N}^{\star}$. Step (C.16) follows from Definition C.4 applied to $names^{\dagger}$. Step (C.17) follows by hypothesis: there is no x such that $(x, s) \in names^{\dagger}$. Step (C.18) follows from IH as if $\mathsf{D}^{\dagger}.name(n) \neq \mathsf{D}^{\star}.name(n)$, then $\mathsf{D}^{\dagger}.name(n) = \bot$ and $(\mathsf{D}^{\dagger}.ni(n), \mathsf{D}^{\star}.name(n)) \in$ *hidden*. Step (C.19) follows from the fact that $\mathsf{D}^{\dagger}_n.ni$ is injective, hence there is only one n such that $\mathsf{D}^{\dagger}_n.ni(n) = ind$.

By definition of $\overline{\mathsf{N}}^{\dagger}(S^{\circ}, \mathsf{D}^{\circ})$, if $(v_2, s) \in$ *hidden* and there is no x such that $(x, s) \in names^{\dagger}$, then no message containing v_2 was ever sent to the adversary. Moreover, $\mathsf{D}^{\dagger}_n.name(n) = \bot$ implies by definition of shadow that n does not occur free in S_n and does not occur in Φ_n nor $\mathsf{D}^{\dagger}_n.certval$. Since n does not occur free in S_n nor Φ_n, and n is a subterm of $M^{\star}$ (v_2 is in $\ulcorner M^{\star} \urcorner^{\mathsf{D}'',\mathcal{H}}$ and $\mathsf{D}^{\star}_n.ni(n) = v_2$ implies that n is a subterm of $M^{\star}$), we cannot have $S_n \xrightarrow{(M^{\star})} S'$, hence the failure flag is set in $\mathsf{N}^{\star}$. We conclude that if $\mathsf{N}^{\dagger}_n$ and $\mathsf{N}^{\star}_n$ unmarshal the message to two different internal representations, $\mathsf{N}^{\star}$ has the failure flag set, hence we are in Case 2.

As for internal reductions, since both machines unmarshal the message to the same internal representation whenever both do not fail, our deterministic scheduler performs the same reductions in both cases.

As for outputs $\mathsf{N}^{\dagger}_{n}$ and $\mathsf{N}^{\star}_{n}$ are also able to produce the same output: either it is a message to $b \in \mathcal{H}$, and in this case $\mathsf{N}^{\dagger}_{n}$ adds a tuple $(\mathit{cert}, \mathit{sig})$ to *hidden* while $\mathsf{N}^{\star}_{n}$ adds it to names, or it is a message to $e \notin \mathcal{H}$ and in this case $\mathsf{N}^{\dagger}_{n}$ uses the same bitstring for the name as $\mathsf{N}^{\star}_{n}$ either because both generate it or because it was fetched from *hidden*. The condition on $\mathsf{D}^{\star}$ is trivially verified. It may also occur a clash on the generation of names. In this case a failure flag is set in $\mathsf{N}^{\star}$ and we are in Case 2. □

By composing these equivalences, we obtain

Lemma C.15. $\overline{\mathsf{M}}(S^{\circ}, \mathsf{D}^{\circ}) \approx \overline{\mathsf{N}}(S^{\circ}, \mathsf{D}^{\circ})$.

Proof. Let $\overline{\mathsf{A}}$ be a PPT adversary, with a run-time bounded by the polynomial $p_{\overline{\mathsf{A}}}(\cdot)$. Let

$$\Delta^{i}_{j}(\eta) = \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(i,\ldots,i-1,i-1,\ldots,i-1)}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] - \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(i,\ldots,i,i-1,\ldots,i-1)}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1]$$

where the tuples differ at position j, $1 \leq j \leq |\mathcal{H}|$. Let

$$f(\eta) = \sum_{i=1}^{p_{\overline{\mathsf{A}}}(\eta)} \sum_{j=1}^{|\mathcal{H}|} \Delta^{i}_{j}(\eta).$$

By Lemma C.12, $\Delta^{i}_{j}(\eta)$ is a negligible function of η for all i, j. It follows that $f(\eta) \leq |\mathcal{H}| \times p_{\overline{\mathsf{A}}}(\eta) \times \max_{1 \leq i \leq p_{\overline{\mathsf{A}}}(\eta), 1 \leq j \leq |\mathcal{H}|} \Delta^{i}_{j}(\eta) = |\mathcal{H}| \times p_{\overline{\mathsf{A}}}(\eta) \times g(\eta)$ where $g(\eta)$ is a negligible function. Applying Proposition A.3 to $g(\eta)$ and $|\mathcal{H}| \times p_{\overline{\mathsf{A}}}(\eta)$ we get that $f(\eta)$ is a negligible function. Now, expanding $f(\eta)$ we obtain that

$$f(\eta) = \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(0,\ldots,0)}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] - \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(p_{\overline{\mathsf{A}}}(\eta),\ldots,p_{\overline{\mathsf{A}}}(\eta))}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1]$$

that is,

$$\Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(0,\ldots,0)}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] - \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(p_{\overline{\mathsf{A}}}(\eta),\ldots,p_{\overline{\mathsf{A}}}(\eta))}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] \leq \operatorname{neg}(\eta). \tag{C.20}$$

By Lemmas C.10 and C.11,

$$\Pr[\overline{\mathsf{A}}[\overline{\mathsf{M}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] - \Pr[\overline{\mathsf{A}}[\overline{\mathsf{N}}^{(0,\ldots,0)}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1] \leq \operatorname{neg}(\eta). \tag{C.21}$$

Since at most $p_{\overline{\mathsf{A}}}(\eta)$ messages are exchanged in any given run and the machines $\overline{\mathsf{N}}^{\widetilde{p_{\overline{\mathsf{A}}}(\eta)}}(S^{\circ}, \mathsf{D}^{\circ})$ and $\overline{\mathsf{N}}^{\widetilde{\omega}}(S^{\circ}, \mathsf{D}^{\circ})$ may have different behaviours only after $p_{\overline{\mathsf{A}}}(\eta)$ exchanges, we have

$$\Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{N}}^{\widetilde{p_{\overline{\mathsf{A}}}(\eta)}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] - \Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{N}}^{\widetilde{\omega}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] = 0. \tag{C.22}$$

By Lemmas C.13 and C.14, we have

$$\Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{N}}^{\widetilde{\omega}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] - \Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{N}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] \leq \operatorname{neg}(\eta). \tag{C.23}$$

Composing (C.21), (C.20), (C.22), and (C.23), we finally obtain

$$\Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{M}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] - \Pr[\ \overline{\mathsf{A}}[\overline{\mathsf{N}}(S^{\circ}, \mathsf{D}^{\circ})] \longrightarrow 1\ ] \leq \operatorname{neg}(\eta).$$

□

C.5 Main Proofs

We are now ready to complete the proofs for our main theorems.

Restatement of Theorem 3.4. *Let S be a safe stable system, D a valid shadow for S, and A a PPT algorithm.*

The probability that $\mathsf{A}[\mathsf{M}(S, \mathsf{D})]$ completes and leaves the system in state M' with $\mathsf{M}' \neq \mathsf{M}(S', \mathsf{D}')$ for any normal transitions $S \xrightarrow{\varphi} S'$ with valid shadow D' is negligible.

Proof. Let S be a safe stable system with valid shadow D and A a PPT algorithm (with no access to the failure flag of $\overline{\mathsf{M}}(S, \mathsf{D})$). By definition of D being a valid shadow for S, there exist a safe initial system S° with initial shadow D°, normal transitions $S^\circ \xrightarrow{\varphi^\circ} S$, and a PPT algorithm $\mathsf{A}_\circ$ such that $\mathsf{A}_\circ[\mathsf{M}(S^\circ, \mathsf{D}^\circ)] \longrightarrow \mathsf{M}(S, \mathsf{D})$.

Let M' *bad* abbreviate that there are no normal transitions $S \xrightarrow{\varphi} S'$ and valid shadow D' for S' such that $\mathsf{M}' = \mathsf{M}(S', \mathsf{D}')$. Let $\mathsf{M}'_\circ$ *bad* (resp. $\overline{\mathsf{M}}'_\circ$ *bad*) abbreviate that there are no normal transitions $S^\circ \xrightarrow{\varphi} S'$ and valid shadow D' for S' such that $\mathsf{M}'_\circ = \mathsf{M}(S', \mathsf{D}')$ (resp. $\overline{\mathsf{M}}'_\circ = \overline{\mathsf{M}}(S', \mathsf{D}')$).

$$\begin{aligned}
& \Pr\left[\mathsf{A}[\mathsf{M}(S, \mathsf{D})] \longrightarrow s_{\mathtt{r}}(\mathsf{M}') \wedge \mathsf{M}' \textit{ bad}\right] && \text{(C.24)}\\
= \; & \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\mathsf{M}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{M}'_\circ) \wedge \mathsf{M}'_\circ \textit{ bad}\right] && \text{(C.25)}\\
= \; & \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\overline{\mathsf{M}}'_\circ) \wedge \overline{\mathsf{M}}'_\circ \textit{ bad}\right] && \text{(C.26)}\\
\leq \; & \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \textit{ fails}\right] \\
& \qquad + \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\overline{\mathsf{M}}'_\circ) \wedge \overline{\mathsf{M}}'_\circ \textit{ bad}\right] && \text{(C.27)}\\
\leq \; & \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \textit{ fails}\right] && \text{(C.28)}\\
\leq \; & \Pr\left[(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{N}}(S^\circ, \mathsf{D}^\circ)] \textit{ fails}\right] + \operatorname{neg}(\eta)\,. && \text{(C.29)}\\
\leq \; & \operatorname{neg}(\eta) + \operatorname{neg}(\eta)\,. && \text{(C.30)}
\end{aligned}$$

The probability (C.24) is the target probability of Theorem 3.4. Step (C.25) applies the definition of initialisation of S with valid shadow D, and Lemma 3.3, $S^\circ \xrightarrow{\varphi^\circ} S_1$ implies that $S_1 \equiv S$. Step (C.26) holds by Definition C.1: since A does not read the flag, the additional checks performed by $\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)$ do not affect the outcome of any given run. Step (C.27) splits the probability depending on the predicate $(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)]$ *fails*; the inequality appears as we ignore $(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\overline{\mathsf{M}}'_\circ)$ and $\overline{\mathsf{M}}'_\circ$ *bad* in case $(\mathsf{A}_\circ; \mathsf{A})[\overline{\mathsf{M}}(S^\circ, \mathsf{D}^\circ)]$ fails. Step (C.28) follows from Lemma C.6 applied to S°, D°, and $(\mathsf{A}_\circ; \mathsf{A})$: the two predicates within the second probability are mutually exclusive, so the second probability is zero. Step (C.29) follows from Lemma C.15 applied to $S^\circ, \mathsf{D}^\circ$ and indistinguishability (Definition C.2) applied to an adversary $\overline{\mathsf{A}}$ that runs $(\mathsf{A}_\circ; \mathsf{A})$ and then returns the failure flag. Step (C.30) applies Lemma C.8 to S°, D°, and $(\mathsf{A}_\circ; \mathsf{A})$. □

Lemma C.16. *Let S_1° and S_2° be safe initial systems with initial shadow D°.*

If $S_1^\circ \simeq S_2^\circ$, then for all output messages $s_{\mathtt{r}}$,

$$\Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N}_1)\right] = \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N}_2)\right].$$

Proof. We first show by induction on n that, for any given run, if

$$\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{N}_1$$

and the failure flag is not set in N_1, then

1. there exist S_1 with shadow D_1 and normal transitions $S_1^\circ \xrightarrow{\varphi} S_1$ such that $\mathsf{N}_1 = \overline{\mathsf{N}}(S_1, \mathsf{D}_1)$, and
2. $\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{N}_2$, the failure flag is not set in N_2 and $\mathsf{N}_2 = \overline{\mathsf{N}}(S_2, \mathsf{D}_2)$ for some $S_2 : S_2^\circ \xrightarrow{\varphi} S_2$, $S_1 \simeq S_2$, and $\mathsf{D}_2 = \mathsf{D}_1$ except for definition of internal identifiers and the associated messages in *enctable* (i.e., $\mathsf{D}_2.ni$ and $\pi_1(\mathsf{D}_2.wire)$). Moreover, if we denote by $N = \{n \in \mathsf{Name} : \mathsf{D}_1.name(n) \text{ is defined}\}$ there is a bijection between $\mathsf{D}_1.ni|_N$ and $\mathsf{D}_2.ni|_N$ where $f|_N$ denotes the restriction of f to domain N.

The lemma follows easily from this property (using symmetry) as the outputs of the two runs are the same, hence all the available information to the adversary is the same in both cases. Let us then prove the property by induction in n.

Base case $n = 0$: We have $\mathsf{N}_1 = \overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)$. We use $S_1 = S_1^\circ, \mathsf{D}_1 = \mathsf{D}^\circ$, and $\varphi = \varepsilon$. The partial function D° is initially undefined so $\mathsf{D}_1.name$ is trivially injective. As for Condition 2, $\mathsf{N}_2 = \overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)$. We use $S_2 = S_2^\circ, \mathsf{D}_2 = \mathsf{D}^\circ$, and $\varphi = \varepsilon$. Trivially $\mathsf{D}_2 = \mathsf{D}_1$.

Inductive case: Suppose that

$$\overline{\mathsf{N}}(S_1, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{N}_1' \overset{(inp_{n+1})}{\rightsquigarrow} \rightsquigarrow^* \overset{s_{n+1}}{\rightsquigarrow} \mathsf{N}_1$$

and that the failure flag is not set in N_1. By induction hypothesis for n,

(a) there exist S_1' with shadow D_1' and normal transitions $S_1^\circ \xrightarrow{\varphi'} S_1'$ such that $\mathsf{N}_1' = \overline{\mathsf{N}}(S_1', \mathsf{D}_1')$, and

(b) $\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ) \overset{(inp_1)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_1}{\rightsquigarrow} \overset{(inp_2)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_2}{\rightsquigarrow} \dots \overset{(inp_n)}{\rightsquigarrow} \rightsquigarrow^* \overset{s_n}{\rightsquigarrow} \mathsf{N}_2'$, the failure flag is not set in N_2' and $\mathsf{N}_2' = \overline{\mathsf{N}}(S_2', \mathsf{D}_2')$ for some $S_2' : S_2^\circ \xrightarrow{\varphi'} S_2'$, $S_1' \simeq S_2'$, and $\mathsf{D}_2' = \mathsf{D}_1'$ except for definition of internal identifiers and the associated messages in *enctable*. Denoting by $N' = \{n \in \mathsf{Name} : \mathsf{D}_1'.name(n) \text{ is defined}\}$ there is a bijection between $\mathsf{D}_1'.ni|_{N'}$ and $\mathsf{D}_2'.ni|_{N'}$.

We have that both S_1' and S_2' are safe (since S_1° and S_2° are) and stable (by definition of normal transitions). Applying Lemma C.7 to our hypothesis we have that there exist S_1 with shadow D_1 and normal transitions $S_1^\circ \xrightarrow{\varphi'} S_1' \xrightarrow{\alpha\widetilde{\beta}} S_1$ such that $\mathsf{N}_1 = \overline{\mathsf{N}}(S_1, \mathsf{D}_1)$ which establishes Condition 1.

To prove Condition 2 one should notice that the keys in the cache of N_1' and N_2' are the same ($\mathsf{D}_1'.keycache = \mathsf{D}_2'.keycache$) so the same messages will be accepted by the two machines. Also, since there is a bijection between $\mathsf{D}_1'.ni|_{N'}$ and $\mathsf{D}_2'.ni|_{N'}$ we have that $names_1' =$

$names'_2$ up to renaming of the internal identifiers ($names'_i$ is the $names$ table for N'_i), and since $\mathsf{D}'_1.certval = \mathsf{D}'_2.certval$ we have also that $signed'_1 = signed'_2$ ($signed'_i$ is the $signed$ table for N'_i).

Now there are two different cases:

(i) if (inp_{n+1}) is a message from the adversary, it will be unmarshaled in both cases to the same internal representation m up to a renaming of internal identifiers for names, hence $S'_1 \xrightarrow{(M)} S''_1$ and $S'_1 \simeq S'_2$ imply that $S'_2 \xrightarrow{(M)} S''_2$ (N'_2 does not fail); by safety, we know that the complexity of these internal reductions is bound by $p_{S_1}(\lceil M \rceil)$ and $p_{S_2}(\lceil M \rceil)$, which means that both machines terminate and in particular $\mathsf{N}'_2 \overset{(inp_{n+1})}{\rightsquigarrow}\rightsquigarrow^*\overset{s'}{\rightsquigarrow} \mathsf{N}_2$. As for the output $S''_1 \xrightarrow{\beta} S_1$, hence $S''_2 \xrightarrow{\beta} S_2$ for some S_2 and $S_1 \simeq S_2$. As the only marshaled names are the names in messages to the adversary, $\mathsf{D}'_1.name = \mathsf{D}'_2.name$, and the names sent to the adversary are the same in both cases, the failure flag is not set in N_2. As for messages to honest users, they may differ from S''_1 to S''_2 but by definition both N'_1 and N'_2 only encrypt zero's which imply that $s' = s_{n+1}$. Considering the shadow D_2 defined as $\mathsf{D}_2.ni = \mathsf{D}'_2.ni$ plus all the identifiers generated during the reduction, $\mathsf{D}_2.wire = \mathsf{D}_1.wire$ except that the first component includes the message that was sent by N_2 and all the other components of D_2 as D_1, we have that $\mathsf{N}_2 = \overline{\mathsf{N}}(S_2, \mathsf{D}_2)$, and there exists a bijection between $\mathsf{D}_1.ni|_N$ and $\mathsf{D}_2.ni|_N$ where $N = \{n \in \mathsf{Name} : \mathsf{D}_1.name(n) \text{ is defined}\}$.

(ii) if (inp_{n+1}) is a message from an honest user, and M/i is defined in S'_1 with $\mathsf{D}'_1.wire(i) = \_, k, \_, \_$, then there is also an input (i) in S'_2 and since $\mathsf{D}'_2.wire = \mathsf{D}'_1.wire$ except for the first component, N'_2 will not fail on this input. Regardless to what this input is unmarshaled, if $S'_1 \xrightarrow{(i)\tilde{\beta}} S_1$ then $S'_2 \xrightarrow{(i)\tilde{\beta}} S_2$ for some stable S_2. The rest of the argument is similar to the previous case.

Our claim follows directly from this property. □

Lemma C.17. *Let S_1° and S_2° be safe initial systems with initial shadow* D°.
If $S_1^\circ \simeq S_2^\circ$, then $\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ) \approx \overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)$.

Proof. Let $\overline{\mathsf{A}}$ be an arbitrary PPT algorithm. We have that

$$\begin{aligned}
& \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{N}_1)\right] && \text{(C.31)}\\
\leq\ & \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)]\ \textit{fails}\right] + \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N}_1)\right] && \text{(C.32)}\\
\leq\ & \operatorname{neg}(\eta) + \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N}_1)\right] && \text{(C.33)}\\
=\ & \operatorname{neg}(\eta) + \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)] \xrightarrow{+} s_{\mathtt{r}}(\mathsf{N}_2)\right] && \text{(C.34)}\\
\leq\ & \operatorname{neg}(\eta) + \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{N}_2)\right] && \text{(C.35)}
\end{aligned}$$

Step (C.32) splits the probability depending on the predicate $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)]$ *fails*; the inequality appears as we ignore $\mathsf{A}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{N}_1)$ when $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)]$ fails. Step (C.33) follows from Lemma C.8 applied to S_1°, D°, and $\overline{\mathsf{A}}$. Step (C.34) follows from Lemma C.16. Step (C.35) follows from the inclusion of all cases where $\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)]$ fails in the second probability. By symmetry, we conclude

$$|\Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{N}_1)\right] - \Pr\left[\overline{\mathsf{A}}[\overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)] \longrightarrow s_{\mathtt{r}}(\mathsf{N}_2)\right]| \leq \operatorname{neg}(\eta)\,.$$

□

We are now ready to prove our theorem about soundness of equivalences.

Restatement of Theorem 3.5. *Let S_1 and S_2 be safe stable systems,* D *a valid shadow for both S_1 and S_2.*

If $S_1 \simeq S_2$, then $\mathsf{M}(S_1, \mathsf{D}) \approx \mathsf{M}(S_2, \mathsf{D})$.

Proof. By Lemma 3.3, since $S_1 \simeq S_2$, there exist safe initial systems S_1°, S_2° and labels φ° such that $S_1^\circ \simeq S_2^\circ$, $S_1^\circ \xrightarrow{\varphi^\circ} S_1$, and $S_2^\circ \xrightarrow{\varphi^\circ} S_2$. Applying Lemma C.17 to S_1° and S_2° we obtain that $\overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)$.

Applying Lemma C.15 to S_1°, S_2° and D°, we have $\overline{\mathsf{M}}(S_1^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{N}}(S_1^\circ, \mathsf{D}^\circ)$ and $\overline{\mathsf{M}}(S_2^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{N}}(S_2^\circ, \mathsf{D}^\circ)$. By transitivity, we obtain

$$\overline{\mathsf{M}}(S_1^\circ, \mathsf{D}^\circ) \approxeq \overline{\mathsf{M}}(S_2^\circ, \mathsf{D}^\circ). \tag{C.36}$$

By Definition C.1, an adversary A able to distinguish $\mathsf{M}(S_1^\circ, \mathsf{D}^\circ)$ from $\mathsf{M}(S_2^\circ, \mathsf{D}^\circ)$ also distinguishes $\overline{\mathsf{M}}(S_1^\circ, \mathsf{D}^\circ)$ from $\overline{\mathsf{M}}(S_2^\circ, \mathsf{D}^\circ)$, so (C.36) implies

$$\mathsf{M}(S_1^\circ, \mathsf{D}^\circ) \approx \mathsf{M}(S_2^\circ, \mathsf{D}^\circ). \tag{C.37}$$

We finally show that $\mathsf{M}(S_1, \mathsf{D}) \approx \mathsf{M}(S_2, \mathsf{D})$ by considering the initialisation protocol $\mathsf{A}_\circ$. If there is an adversary A that can distinguish $\mathsf{M}(S_1, \mathsf{D})$ from $\mathsf{M}(S_2, \mathsf{D})$, then the adversary $(\mathsf{A}_\circ; \mathsf{A})$ is able to distinguish $\mathsf{M}(S_1^\circ, \mathsf{D}^\circ)$ from $\mathsf{M}(S_2^\circ, \mathsf{D}^\circ)$, and this contradicts (C.37). □