Modelling and analyzing security protocols in cryptographic process calculi

Steve Kremer

Modelling and analyzing security protocols in cryptographic process calculi

Mémoire d'habilitation à diriger des recherches

Steve Kremer

Commitee:

- Martín Abadi (rapporteur)
- Ran Canetti (rapporteur)
- Hubert Comon-Lundh
- Jean-Pierre Jouannaud
- Catuscia Palamidessi (rapporteur)
- David Pointcheval
- Michael Rusinowitch
- Andre Scedrov

Contents

CHAPTER 1

Introduction

Preamble. This habilitation thesis summarizes a selection of my research results obtained since my PhD thesis. This summary is not exhaustive. I will not discuss [38], a result in the vein of my PhD thesis, or results of [36, 30], using different formalisms and techniques than most results presented here. Nor will I discuss my results in the field of mathematics of juggling [39, 14]. By convention, citations of my own work will use a plain, numerical style, *e.g.* [1], while I will refer to other work using alpha-numerical style, *e.g.*, [ABB+05].

1.1 Security protocols and the need for formal verification

1.1.1 Security protocols everywhere!

Security protocols are distributed programs that aim at establishing some security properties, *e.g.*, confidentiality of some data, authentication or anonymity, over an untrusted network, *e.g.*, the Internet. To achieve these properties the protocols generally use *cryptography* to encrypt or digitally sign messages. A typical example of such a protocol is the Transport Layer Security (TLS) protocol and its predecessor, the Secure Sockets Layer (SSL). These protocols are for instance used for guaranteeing a secure connection to a web site in particular for secure payment over the Internet. Most web browsers display a small lock to indicate that you are executing a secure session using one of these protocols. As A. Mercier reported in his PhD thesis [Mer09]: according to the "Fédération du e-commerce et de la vente à distance" (federation for e-commerce and remote selling) in 2008 in France 78% of French people use remote selling, 80% of these sales use the Internet. Another example is the use of mobile phones which need to authenticate to the phone operator (to avoid someone else passing phone calls on your account). Yet another emergent example of a security protocol is electronic voting with its obvious security risks: in March 2011, in Estonia (for the sixth time) as well as in Switzerland, Internet voting will be available for the parliamentary elections (even though in Switzerland the number of voters with access to this mode of voting is not yet known); in Norway, at the end of 2011 municipal and county elections will be held and legally binding internet voting will be available in selected municipalities and for selected voter groups [Kri10].

1.1.2 The need for formal methods

It has been shown that security protocols are notoriously error-prone, even on protocols of moderate size. To illustrate this we give three examples.

- In 1995, for instance, Lowe [Low95] was able to find a flaw[1] in the Needham-Schroeder public key protocol [NS78], 17 years after its publication. This attack has become a kind of "benchmark" example for automated tools.
- More recently, the SAML 2.0 Web Browser Single Sign-On authentication system developed by Google has been attacked. The Single Sign-On protocol allows a user to identify himself only once and then access to various applications (such as Gmail or Google calendar). While designing a formal model of this protocol, Armando *et al.* [ACC+08] discovered that a dishonest service provider could actually impersonate any of its users at another service provider. This flaw has been corrected since.
- Using fully automated tools, Bortolozzo *et al.* [BCFS10] were able to find serious security flaws in several commercial tokens implementing the PKCS#11 Cryptoki standard.

Such flaws can have serious economic (or political) consequences. To avoid them, security protocols must be built on solid foundations and rigorous proofs of security are needed. Formal methods such as model-checking and automated reasoning have shown very effective for analyzing critical systems. As we will see in the following section, a variety of formal techniques have been applied to the analysis of security protocols.

1.1.3 An example of a security protocol

For illustration purposes we present a simple handshake protocol (which was also used in Blanchet's habilitation thesis [Bla08] for the same purpose). The protocol is executed between an initiator A and a responder B. The protocol's goal is that A and B share a secret s at the end of the execution. The honest execution of the protocol is depicted as in Figure 1.1. A first generates a fresh symmetric session key k and signs this key with his private signature key ska, yielding the message $\mathsf{sign}(k, ska)$. Then A uses an asymmetric encryption scheme to encrypt the signed key with the public key corresponding to B's private decryption key: $\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(dkb))$. The rationale of this message is as follows: when B receives such a message, A's signature ensures that the message originated from A and the encryption is expected to ensure the confidentiality of the key k (as only B can decrypt this message). B can now use the secret key k to symmetrically encrypt s and send return this message to A.

As already mentioned, in Figure 1.1, we only represent the intended execution of the protocol, which is often misleading. As we will see later, a better representation is to model the two *roles* separately, as two communicating processes. Indeed this naïve handshake protocol admits a *man-in-the-middle* attack, which we describe in Figure 1.2: if A initiates a session with a malicious entity C, then C is able to fool the responder B and make B believe that he shares the secret s with A, while it is actually shared with C.

[1] It is arguable whether Lowe's attack is in the scope of the initial hypothesis [NS78], but the attack should certainly be considered in the context of an open network.

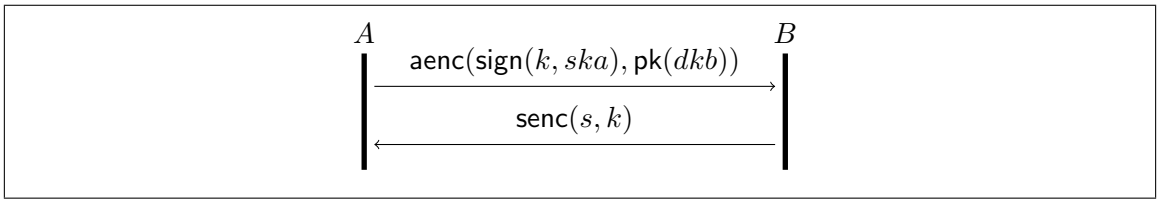


Figure 1.1: (Flawed) handshake protocol

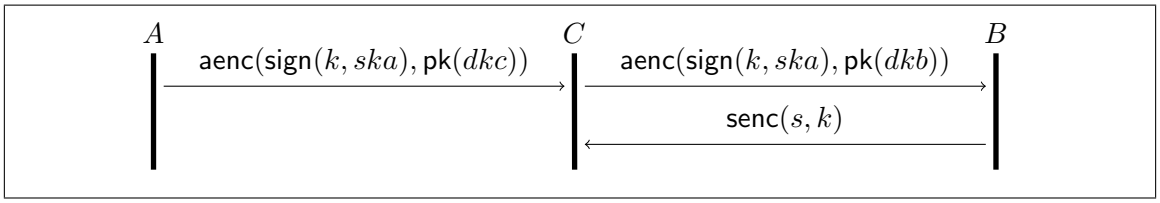


Figure 1.2: Man in the middle attack

This attack is possible because there is no link between A's signature on k and the entity whose key is used for the encryption. The usual way to avoid this kind of attacks (also suggested by Lowe [Low96]) is to include the participant's identities in the messages. A corrected version of the protocol, adding the identities A and B to the signature is presented in Figure 1.3.

Many more examples of protocols can be found in a survey by Clark and Jacob [CJ97], in the Security Protocols Open Repository (SPORE) [SPO] and the AVISPA collection of security protocols [AVI].

1.2 Some existing results

The use of symbolic methods for analyzing security protocols goes back to Needham and Schroeder [NS78] and in particular to the seminal paper of Dolev and Yao [DY81]. The "ingredients" of symbolic methods can be summarized as follows. Messages are represented using an abstract term algebra. While the adversary is computationally unbounded it is however only allowed to manipulate messages according to a predefined set of rules. Moreover, random numbers are represented by fresh names and collisions are impossible. The adversary is supposed to have full control of the network: any message that is output is sent to the adversary, who can replay, modify or forge new messages, according to his current knowledge. Moreover, the adversary can initiate an unbounded number of sessions with honest protocol participants. This kind of model contrasts with computational models where adversaries are arbitrary probabilistic polynomial-time Turing machines and mes-

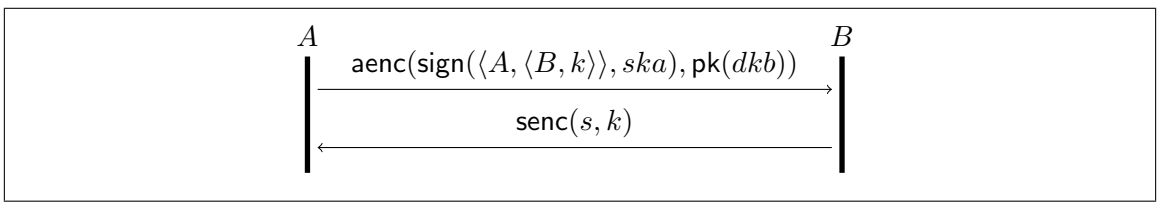


Figure 1.3: Corrected handshake protocol

sages are bitstrings without any particular structure. Hence, random numbers are drawn from a finite set of bitstrings making collisions possible.

In this section we give a short overview of existing work. The aim is certainly not to be exhaustive, but to give an overview of the variety of techniques and results.

1.2.1 What can we decide about security protocols?

Before discussing different analysis techniques we will review some of the decidability and complexity result that have been obtained. We first discuss reachability properties. Even a simple property such as secrecy (modelled as undeducibility) is undecidable in general. Several proofs of undecidability have been presented when the number of sessions is unbounded, *e.g.* in [EG83, DLM04, CLC05, Ara08]. Undecidability can even be shown when we bound the size of messages [DLM04].

There exist however particular decidable subclasses. Rusinowitch and Turuani [RT01] have shown that deciding secrecy is coNP-complete when the number of sessions is bounded. This first result was obtained for protocols that only use encryption. These results have been lifted to protocols that use exclusive or [CKRT03b] and Diffie-Hellman exponentiation [CKRT03a].

There also exist decidable subclasses of protocols in the case of an unbounded number of sessions: an early result is the PTIME complexity result by Dolev *et al.* [DEK82] for a restricted class, called *ping-pong* protocols. Other classes have been proposed by Ramanujam and Suresh [RS03, RS05], and Lowe [Low98]. However, in both cases, temporary secrets, composed keys and ciphertext forwarding are not allowed which discards protocols, such as the Yahalom protocol [CJ97]. More recently, we [26] have shown for a class of tagged protocols, that these restrictions can be avoided.

For indistinguishability, much less results are known. As one may expect, the problem is undecidable as soon as the number of sessions is unbounded. Hüttel [Hüt02] has also shown that in the Spi calculus [AG99], framed bisimilarity (modelling indistinguishability in the presence of an active adversary) is undecidable for the finite control fragment. Hüttel also presents a decision procedure for a bounded number of sessions (the fragment without replication). However, the procedure has multi-exponential time complexity. A subproblem for deciding indistinguishability of processes is to decide whether two sequences of terms are indistinguishable. This problem, formalized as static equivalence, is decidable for a large number of equational theories [AC06, CD07, BCD09, 6].

1.2.2 Models for security protocols and proof techniques

Process algebras. Process algebras provide a natural modelling of protocols where each role of a protocol corresponds to an independent process. Process algebras such as CSP have been extensively used for modelling and analysing security protocols [RSG+00]. Lowe designed an analysis tool, Casper [Low97], built on top of the finite model-checker FDR. Schneider and Heather [HS00] used rank functions as a proof technique for showing the security for an unbounded number of sessions. The CCS variant CryptoSPA [FM99] has also been used to analyse security protocols. Abadi and Gordon introduced a variant of the pi calculus, the spi-calculus [AG99] with constructs for particular cryptographic primitives.

The applied pi calculus [AF01] generalizes the spi calculus and allows to model cryptographic primitives using arbitrary equational theories. This yields an accurate (symbolic) modelling of protocols, in particular generation of fresh nonces (using restriction) and the possibility for an unbounded number of sessions by using replication.

The main difference between the different models based on process algebra lies in the definition of security properties. Many models [AL00, ALV02a, Bor01] consider reachability properties. Other models [AG99, AF01] base their definition of security properties on an observational equivalence allowing to model a wider range of properties including for instance anonymity properties. In several of these calculi [BDNP99, AG98, Cor02, BN02], coinductive characterization of observational equivalence in terms of a bisimulation have been proposed, yielding powerful (manual) proof techniques. Several symbolic bisimulations have also been proposed for the spi calculus [BBN04, Bor08] and the applied pi calculus [31, 9, LL10].

Trace based models and inductive reasoning. The strand space model [Gut01, THG99] is a model representing the set of possible traces of a protocol execution. The model comes with an appealing graphical representation, and some proof techniques [GT00b]. Protocol composition results have also been obtained [GT00a]. However, the model and its proof techniques strongly rely on the assumption that the message algebra is free and it seems difficult to extend the model with more complex cryptographic primitives.

Paulson [Pau98] also defined a model where the set of protocol traces is defined inductively and used the Isabelle theorem prover to analyze protocols. However, human interaction is needed to guide the theorem prover and many important lemmas used as proof techniques are tightly bound to the particular cryptographic primitives defined in the model. Some key lemmas would for instance be wrong if one allows keys to be constructed rather than atomic.

Tree automata and Horn clauses. Tree automata and Horn clauses (which can encode tree automata) can be used to compute an over-approximation of the attacker knowledge for an unbounded number of sessions. They provide a low-level representation of protocols, when assuming an unbounded number of sessions, which is convenient to manipulate with automated tools. Weidenbach [Wei99] encoded both the protocol and the attacker capabilities in Horn clauses and applied the theorem prover Spass to the verification of security protocols. Genet and Klay [GK00] combined the use of tree automata and rewriting to verify security protocols, which lead to a recent tool TA4SP, which is part of the AVISPA tool suite [ABB$^+$05]. Goubault-Larrecq [Gou05] identified a class of Horn clauses for which resolution is decidable (but still approximating the security of protocols). A recent extension of his tool [Gou08] can be used to automatically derive security proofs in the Coq theorem prover. Blanchet developed Horn clause resolution techniques, dedicated to the verification of security protocols, that have been implemented in the ProVerif tool. This tool has been developed for nearly 10 years and is now one of the state of the art tools which is able to verify confidentiality [Bla01], authentication [Bla09] and even some equivalence properties [Bla04, BAF05] for many cryptographic primitives, specified as an equational theory.

Constraint solving. A symbolic trace of a protocol, *i.e.* an interleaving of roles, can be represented using a constraint system. As Horn clauses can be used as a low level representation when considering an unbounded number of sessions, constraint systems can be thought of as such a representation when the number of sessions is bounded. (Constraint systems are however an exact representation.) For the analysis of security protocols, constraint systems were first used by Shmatikov and Millen [MS01], who proposed a prototype tool based on a constraint solving algorithm. The tool was optimized by Corin *et al.* [CE03] and extended to verify properties in an LTL like logic with past, called PS-LTL [CES06]. Since then more constraint solving algorithms have been proposed in order to handle more cryptographic primitives with algebraic properties [CS03, Shm04] and for verifying particular properties [33, CCZ10]. The probably most mature tool using constraint solving technique is Turuani's CL-Atse protocol analyzer [Tur06] which also provides support for exclusive or and exponentiation.

The problem of verifying trace properties corresponds to the satisfiability of constraint systems. In [Bau05], Baudet noted that the indistinguishability of two symbolic traces corresponds to checking whether two constraint systems have the same set of solutions. He proposes a first procedure for deciding the equivalence of constraint systems. More, recently, a new decision procedure, more suitable for implementation, was proposed by Cheval *et al.* [CCD10] together with a prototype implementation.

Logics for security protocols. Dedicated logics, in the style of Hoare logics, have been proposed for the analysis of security protocols. As for program analysis, assertions about the protocol are propagated according to a set of rules. This type of logics goes back to the famous BAN logic [BAN89] which has known many extensions and variations in the early nineties, *e.g.* [GNY90, SvO94]. Recently, a new effort in this direction has been made by Mitchell *et al.* [DDMR07, RDD+08] with a particular emphasis on compositionality resulting in the Protocol Composition Logic (PCL). This logic allows for assume-guarantee reasoning for parallel and sequential composition.

1.2.3 Automated tools

We discuss here some protocol verification tools. Again, our intention is not to be exhaustive.

The AVISPA tool suite. The AVISPA tool suite [ABB+05] regroups four tools which take a common input language HLPSL (which is compiled to an intermediate format IF, specifying a transition system, taken as input by the tools).

- CL-AtSe (Constraint-Logic-based Attack Searcher) is a tool based on constraint solving techniques. It is able to verify the security of a protocol for a bounded number of sessions and an unbounded message size. It provides support for some algebraic properties such as exclusive or, exponentiation and associative pairing and is able to verify secrecy and authentication properties.

- OFMC (On-the-fly Model-Checker) is a state exploration tool performing bounded verification. The tool allows to specify algebraic properties using a language of oper-

ator properties which are then handled using rewriting techniques. However, termination is only obtained by specifying bounds on the message depth and the number of intruder operations used to create new terms. Due to these bounds the tool cannot guarantee completeness.

- SATMC (SAT-based Model-Checker) translates the protocol and the property to verify into a propositional formula which can be checked using efficient SAT solvers. However, only a bounded unrolling of the transition system specified by the IF input is considered.

- TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) analyses an unbounded number of sessions by approximating the intruder knowledge by a regular tree language. If the secret is not included in the language, we can conclude the security of the protocol. TA4SP can also under-approximate the intruder knowledge in order to prove that a protocol is flawed.

ProVerif. ProVerif [Bla01] is a tool for analyzing security protocols without bounding the number of sessions nor the message size. It is based on dedicated Horn clause resolution strategies. The tool is incomplete in the sense that it may fail to prove the security of a protocol, *i.e.* the tool may find false attacks. Moreover, the tool may not terminate (except for a subclass of tagged protocols [BP03] for which termination is ensured). However, the tool performs extremely well in practice. The input language of ProVerif is the applied pi calculus which is then compiled into Horn clauses. Cryptographic primitives may be specified by an arbitrary convergent equational theory (even though the tool may not terminate or fail to prove the convergence of too complicated theories). One of the strong points of ProVerif is that it is able to prove reachability properties, such as deduction based secrecy and (injective) correspondence properties [Bla09], as well as equivalence properties [Bla04, BAF05]. The tool has demonstrated its applicability on a large number of case studies, *e.g.* [ABF07, BC08, BHM08, 37, 19]

Scyther. Scyther [Cre08a, Cre08b] is a recent tool for verifying protocols. The tool does not use any approximation and hence it does not produce false attacks. When the verification fails a counter-example can always be produced. Moreover, the tool is guaranteed to terminate: whenever it is unable to establish unbounded verification, it establishes a form of bounded verification. The theory of Scyther shares a lot of similarities with the strand space model, in particular with [DGT07].

Maude-NPA. Meadows' NRL Protocol Analyzer (NPA) [Mea96] was one of the first tools which was able to prove the correctness of security protocols without bounding the number of sessions or the message size. The tool is based on rewriting techniques and uses a backward search to determine whether a set of "bad" states is reachable. Maude-NPA [EMM09] is the successor of the original NPA and has theoretical foundations in rewriting logic and narrowing. The tool supports equational theories, including associative-commutative (AC) operators, building in particular on the finite variant property [CD05].

1.2.4 The computational approach

Since the 1980s, two distinct communities have analyzed secure protocols. On the one hand symbolic, or Dolev-Yao, methods, as the ones described above, have been developed with the aim of designing automated tools. These models take a rather simplistic view of cryptography, in the sense that the possible attacker actions are explicited. On the other hand, computational models consider adversaries which are arbitrary probabilistic polynomial Turing machines. While such models are arguably more realistic, proofs are also more complex and more difficult to automate. In their seminal paper, Abadi and Rogaway [AR02] have shown that it is possible to bridge these two worlds and "get the best of both": automated proofs which imply the existence of a security proof in a computational model. In [AR02], Abadi and Rogaway considered a setting where the adversary is a pure eavesdropper and only symmetric encryption is used. As they have shown even in such a rather simple setting such proofs are not straightforward, as subtle differences exist between these two settings, (*e.g.*, problems related to key cycles and ciphertexts revealing the length of plaintexts). Since this first paper, many extensions have been proposed, considering more cryptographic primitives, *e.g.*, [BLMW07, 11, 12] and stronger (adaptive [MP05, 32] and active [BPW03, MW04, CC08]) adversaries. Another direction consists in developing automated tools which perform proofs directly in a computational model [Bla06]. We refer the reader to our recent survey [7] for an extensive bibliography on this topic.

1.3 Contributions and outline

1.3.1 A summary of my contributions

My PhD thesis was on the design [46, 47] and analysis [45, 42, 40, 17, 15] of fair exchange protocols. I proposed the use of a temporal logic with game semantics in order to express some of the properties of fair exchange protocols in terms of strategies. In particular this allowed to express, in a natural way, complex properties such as abuse-freeness in optimistic contract signing protocols [42]. The analysis of protocols was automated using the finite state model-checker Mocha and allowed among others to find a fundamental flaw in a multi-party contract signing protocol [40, 15]. Since then I got interested in many areas related to the formal analysis of security protocols.

Modelling and analyzing particular security properties

Electronic voting protocols. During my post-doc at the University of Birmingham I initiated a study of electronic voting protocols together with M. Ryan [37]. At that time we identified several challenges that arised when analysing e-voting protocols: such protocols use non-standard cryptographic primitives which are not treated by most of the existing formalisms, the required properties are non trivial to model and the protocols themselves are complex (several authorities, need for synchronization points, ...) The applied pi calculus [AF01] seemed to be a good choice for a modelling language as it allows to model cryptographic primitives in a general way by the means of an equational theory, offers a rich language for specifying the protocols and allows to easily specify reachability as well as indistinguishability properties. In our first paper [37] we analysed the protocol by Fujioka

et al. [FOO92] covering a series of properties, such as fairness (no early results), eligibility (only eligible voters have the right to vote) and anonymity or vote privacy (votes and voters cannot be linked). However, electronic voting requires stronger privacy properties than merely anonymity. To avoid vote selling and coercion one requests that voters cannot prove, even if he wishes to, how he voted. In collaboration with S. Delaune and M. Ryan [34, 13], we have formalized these notions in the applied pi calculus and provided the first symbolic definitions for these properties. We analyzed several electronic voting protocols and our analysis emphasizes which election administrators need to be trusted in order to ensure them. A summary of our work on privacy properties has been published as an invited chapter in a special LNCS volume dedicated to electronic voting [1]. Yet another important property for electronic elections is *verifiability*: individual voters should be able to check that their vote has been counted in the outcome and any observer should be able to check that the tally is correct. In collaboration with M. Ryan and B. Smyth [20] we have formalized these properties and analysed several protocols (FOO'92 [FOO92], Helios 2.0 [Adi08] and JCJ/Civitas [JCJ05, CCM08]). Again, our approach highlightens which are the exact part of the voting system that need to be trusted in order to achieve the properties.

Security APIs. As viruses and malware may infect computers, sensitive data such as keys should not be stored in a memory which may be accessed by other applications. One way of preventing this is to use tamper proof cryptographic hardware to manipulate such data. PKCS#11 defines an API (Application Programming Interface) for cryptographic devices that has been widely adopted in industry. However, it has been shown to be vulnerable to a variety of attacks that could, for example, compromise the sensitive keys stored on the device. When G. Steel joined our research group I started working on security APIs. In [28, 10] we propose a formal model of the operation of this API. One difference with classical models of security protocols is that it accounts for non-monotonic mutable global state, *i.e.* some parts of the state can be accessed and modified by different sessions. We give decision results for this model and implement the resulting decision procedure using the model checker NuSMV. We report some new attacks and prove the safety of some configurations of the API in our model. We also analyse proprietary extensions proposed by nCipher (Thales) and Eracom (Safenet), designed to address the shortcomings of PKCS#11. Another hardware module which is included in most modern laptops is the *Trusted Platform Module* (TPM). The TPM provides a way to store cryptographic keys and other sensitive data in its shielded memory. Through its API, one can use those keys to achieve some security goals. Although the TPM specification [RSA04] consists of more than 700 pages, the expected security guarantees are not clearly stated. In [19] we model several TPM commands, concentrating on the authentication mechanisms, and identify security properties the TPM should guarantee. We argue that the properties we identified are the ones intended given the security mechanisms that are deployed. Moreover, these properties are indeed the ones that were violated in some known attacks. Using ProVerif we were able to rediscover these known attacks and some new variations on them. We also discuss some fixes to the API, and prove our security properties for the modified API.

Automated verification

Automated verification of equivalence properties. While studying privacy properties of e-voting protocols I realized that the tool support for deciding equivalence properties was insufficient. While the tool ProVerif is able to prove some equivalences [BAF05], there were two reasons why we could not use it to automate privacy proofs on our case studies: *(i)* the equivalence proven by ProVerif is too fine and *(ii)* the equational theories used to model complex cryptographic primitives are beyond the scope of what ProVerif can prove. To overcome the first limitation, I suggested to develop a symbolic semantics for the finite (replication free) fragment of the applied pi calculus. The aim was to to reduce verification of observational equivalence to the equivalence of constraint systems (which was shown decidable by M. Baudet for an important class of equational theories). With S. Delaune and M. Ryan [31, 9] we defined symbolic semantics and a symbolic bisimulation for the applied pi calculus. We have shown this symbolic bisimulation to be sound even though not complete. Nevertheless, our symbolic bisimulation has shown to be "complete enough" to show privacy in e-voting properties. To cover more complex equational theories with Ş. Ciobâcă and S. Delaune we designed a procedure [23, 6] that decides static equivalence for many equational theories, including some theories that were not known to be decidable before. Ş. Ciobâcă also implemented this procedure yielding an efficient prototype. An early version of this work was part of Ş. Ciobâcă's master thesis and the latest version will be part of his PhD thesis. With A. Mercier and R. Treinen, we investigated combination results of non-disjoint equation theories [25, 8] for the decision of static equivalence. In particular we could use this result to decide static equivalence for a theory of bilinear pairing. This work was part of A. Mercier's PhD thesis [Mer09].

Group protocols. In [30] we investigate automated verification of a particular type of security protocols, called group protocols, in the presence of an eavesdropper, i.e., a passive attacker. The specificity of group protocols is that the number of participants is not bounded. Our approach consists in representing an infinite set of messages exchanged during an unbounded number of sessions, one session for each possible number of participants, as well as the infinite set of associated secrets. We use so-called visibly tree automata with memory and structural constraints (introduced recently by Comon-Lundh *et al.* [CJP07]) to represent over-approximations of these two sets. We identify restrictions on the specification of protocols which allow us to reduce the attacker capabilities guaranteeing that the above mentioned class of automata is closed under the application of the remaining attacker rules. The class of protocols respecting these restrictions is large enough to cover several existing protocols, such as the GDH family, GKE, and others. This work was part of A. Mercier's PhD thesis [Mer09].

Compositional reasoning and refinement

As protocols are getting more and more complex and are used as parts of large systems it is interesting to reason about them in a modular and compositional way. We would like to analyze protocols in isolation and conclude that they are secure in an arbitrary environment when executed in the presence of other protocols. It is not difficult to model show that in symbolic models security is preserved when protocols do not share any secrets.

Therefore, one might want to never use the same keying material for different protocols. This is however unrealistic when secrets are user-chosen passwords. Typically, a user will reuse the same password for different protocols. With S. Delaune and M. Ryan [27] we have studied the compositionality of resistance against offline dictionary attacks for password based protocols. We show that resistance against offline dictionary attacks indeed composes in the case one considers a passive adversary, but not in the presence of an active adversary. We show that a simple transformation, where the password is tagged by a protocol identifier using a hash function, both preserves resistance against offline dictionary attacks and guarantees composition even if the same password is reused in different protocols. Another situation where the same keying material is reused is when we execute different sessions of a same protocol which uses long-term keys. With M. Arapinis and S. Delaune [26], we have studied this kind of self-composition: the aim is to show that when a protocol is secure for one session it is also secure when composed with itself in many sessions. While this is obviously not true in general we have designed a compiler which adds a preamble to the protocol where participants agree on a dynamically created tag included in each encryption, signature and hash. For this class of compiled protocols, we show that confidentiality properties are preserved under self-composition. This result can also be interpreted as a class of protocols for which confidentiality is decidable for an unbounded number of sessions. This work was part of M. Arapinis' PhD thesis.

In computational models, universal composability [Can01] (UC) and reactive simulatability [BPW07] (RSIM) were designed to achieve composability. Another aspect of the UC and RSIM frameworks is that ideal functionalities, which one can think of as specifications, can be refined while preserving security protocols. With S. Delaune and O. Pereira [24] we have designed a similar framework for the applied pi calculus. While at first we thought that this would be a straightforward exercise, with the aim of better understanding the corresponding computational models, it turned out to be a non trivial task. In particular the concurrency model of the applied pi calculus differs significantly from the sequential scheduling mechanisms in computational models. Changing the concurrency model raised several interesting questions and led to different design choices.

Computational soundness

Since the 1980s, two approaches have been developed for analyzing security protocols. One of the approaches relies on a computational model that considers issues of complexity and probability. This approach captures a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. The other approach—which my previous work is part of—relies on a symbolic model of protocol executions in which cryptographic primitives are treated as "black boxes", in the sense that the attacker can only perform a given set of actions. Since the seminal work of Dolev and Yao, it has been realized that this latter approach enables significantly simpler and often automated proofs. Unfortunately, the high degree of abstraction and the limited adversary power raise questions regarding the security offered by such proofs. Potentially, justifying symbolic proofs with respect to standard computational models has tremendous benefits: protocols can be analyzed using automated tools and still benefit from the security guarantees of the computational model. In their seminal work, Abadi and Rogaway [AR02] prove the computational soundness of formal (symmetric) encryption in the case of a passive attacker. Since then, many results have been

obtained (see [7] for our recent survey on this topic). Each of these results considers a fixed set of primitives, for instance symmetric or public key encryption. In [35, 12], we design a general framework for comparing formal and computational models in the presence of a passive attacker. We define the notions of soundness and faithfulness of a cryptographic implementation with respect to equality, static equivalence and (non-)deducibility and present proof techniques for showing soundness results illustrated on several examples (xor, ciphers and lists). In [32] we generalise this framework to consider an adaptive adversary, which is strictly more powerful than a passive one. Computational and symbolic models also differ on the way security properties are specified. The standard symbolic deducibility-based notions of secrecy are in general insufficient from a cryptographic point of view, especially in the presence of hash functions: indeed given $\mathsf{h}(s)$ the secret s is not deducible, while a real-or random style, computational definition would trivially not be satisfied if the attacker is presented with either the real secret s or a fresh random s'. In [33] we consider an active adversary and devise a more appropriate secrecy criterion which exactly captures a standard cryptographic notion of secrecy for protocols involving public-key encryption and hash functions: protocols that satisfy this criterion are computationally secure while any violation of our criterion directly leads to an attack.

1.3.2 Outline

The thesis is organized as follows. In Chapter 2, we present notations and basic definitions on term algebras, as well as the applied pi calculus that we use for modelling protocols (for most of our results). We also illustrate the modelling of some security properties in that model. In the following two chapters (Chapters 3 and 4) we review our results on modelling and analyzing properties of particular types of protocols: electronic voting protocols and security APIs. In Chapter 5, we summarize our work on automated methods for deciding equivalence properties. In Chapter 6 we discuss our work on protocol composition and refinement, allowing analysis of protocols in a modular way. Then, in Chapter 7 we summarize our work on computational soundness with the aim of using symbolic techniques to obtain computational proofs. In Chapter 8 we conclude and give directions for future work.

1.3.3 Collaborators

The content of this thesis is based on joint work with the following persons:

- Myrto Arapinis, University of Birmingham, UK
- Mathieu Baudet, MLstate, France
- Rohit Chadha, LSV, France
- Ştefan Ciobâcă, LSV, France
- Véronique Cortier, LORIA, France
- Stéphanie Delaune, LSV, France
- Ralf Küsters, University of Trier, Germany

- Laurent Mazaré, Goldman Sachs, UK
- Antoine Mercier, Ministry of Defense, France
- Olivier Pereira, University of Louvain, Belgium
- Mark D. Ryan, University of Birmingham, UK
- Ben Smyth, LORIA, France
- Graham Steel, LSV, France
- Ralf Treinen, PPS, France
- Bogdan Warinschi, University of Bristol, UK

A complete list of all my co-authors can be found on page 131. In particular some of these works have been done in collaboration with students that I had the pleasure to advise.

- Antoine Mercier defended his PhD thesis [Mer09] entitled "Contributions à l'analyse automatique des protocoles cryptographiques en présence de propriétés algébriques : protocoles de groupe, équivalence statique" in December 2009. The thesis was co-advised by Ralf Treinen. In particular the results described in Section 5.2 were part of Antoine's thesis.
- Ştefan Ciobâcă is scheduled to defend his PhD thesis in fall 2011. This thesis is co-advised by Véronique Cortier. In particular the results described in Section 5.1 will be part of Ştefan's thesis. I also co-advised Ştefan's Master thesis with Stéphanie Delaune.

Additionally, Myrto Arapinis was in 1998 at LSV on a one year temporary research and teaching position to finish her PhD thesis. I co-advised her (with Stéphanie Delaune) during that year. I also had the chance to work with several post-doctoral fellows at LSV: Laurent Mazaré, Graham Steel, Joe-Kai Tsay and Céline Chevalier.

Moreover, part of the future work described in 8.1 is Robert Künnemann's PhD topic. Robert Künnemann started his PhD in October 2010 and is co-advised by Graham Steel.

CHAPTER 2

Preliminaries: messages, protocols and properties

In this chapter we introduce some preliminary definitions and notations. We first introduce basic concepts related to how messages are modelled and manipulated: term algebras, equational theories, deduction and static equivalence. Then, we introduce the applied pi calculus which we will use to model protocols. Finally, we show how several kinds of security properties can be modelled in the applied pi calculus. A reader familiar with these notions can safely skip this chapter.

2.1 Term algebras

A *sorted signature* $(\mathcal{S}, \mathcal{F})$ is defined by a set of *sorts* $\mathcal{S} = \{s, s_1, s_2, \dots\}$ and a set of function symbols $\mathcal{F} = \{f, f_1, f_2, \dots\}$ with arities of the form $\mathit{arity}(f) = s_1 \times \cdots \times s_k \to s$ where $k \geq 0$. If $k = 0$ the symbol is called a *constant* and its arity is simply written s. When $\mathcal{S}$ is a singleton we say that the algebra is *unsorted* and we simply refer to the signature as $\mathcal{F}$. In that case we write $\mathit{arity}(f) = k$ for $\mathit{arity}(f) = s_1 \times \cdots \times s_k \to s$.

We fix an infinite set of sorted *names* $\mathcal{N}$ and an infinite set of *variables* $\mathcal{X}$. The set of *terms of sort* s is defined inductively by :

t	::=		term of sort s
	|	x	variable x of sort s
	|	n	name n of sort s
	|	$f(t_1, \dots, t_k)$	application of symbol $f \in \mathcal{F}$

where each t_i is a term of sort s_i and $\mathit{arity}(f) = s_1 \times \cdots \times s_k \to s$. The set of terms $\mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ is the union of the sets of terms of sort s for every $s \in \mathcal{S}$. We denote by $\mathit{sort}(t)$ the sort of term t. We write $\mathit{vars}(t)$ and $\mathit{names}(t)$ for the set of variables and names occurring in t, respectively. A term t is *ground* iff $\mathit{vars}(t) = \emptyset$. The set of ground terms is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{N})$.

The *positions* $\mathsf{Pos}(t)$ of a term t are defined as usual by $\mathsf{Pos}(u) = \{\epsilon\}$ when $u \in \mathcal{N} \cup \mathcal{X}$ and $\mathsf{Pos}(f(t_1, \dots, t_n)) = \{\epsilon\} \cup \{i \cdot \pi \mid 1 \leq i \leq n, \pi \in \mathsf{Pos}(t_i)\}$ otherwise. The subterm of t

at position p is written $t|_p$, and the replacement in t at position p by u is written $t[u]_p$.

A *substitution* σ written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ (or sometimes $\{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$) with domain $\mathsf{dom}(\sigma) = \{x_1, \ldots, x_n\}$ is a mapping from $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ to $\mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$. We only consider *well sorted* substitutions in which x_i and t_i have the same sort. A substitution σ is *ground* if all t_i are ground. We homomorphically extend the substitution to apply on terms rather than only variables. The *application* of a substitution σ to a term t is written $t\sigma$.

2.2 Equational theories and rewriting systems

An *equation* is an equality $t = u$ where t and u are two terms of the same sort. An *equational theory* $\mathcal{E}$ is a finite set of equations. We denote by $=_\mathcal{E}$ the smallest congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ such that $t\sigma =_\mathcal{E} u\sigma$ for any $t = u \in \mathcal{E}$ and for any substitution σ.

A *term rewriting system* $\mathcal{R}$ is a finite set of *rewrite rules* $l \to r$ where $l \in \mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ and $r \in \mathcal{T}(\mathcal{F}, \mathcal{N}, \mathit{vars}(l))$. A term $u \in \mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ rewrites to v by $\mathcal{R}$, denoted $u \to_\mathcal{R} v$ if there is a rewrite rule $l \to r \in \mathcal{R}$, a position $p \in \mathsf{Pos}(u)$ and a substitution σ such that $u|_p = l\sigma$ and $v = u[r\sigma]_p$. We write $\to^*_\mathcal{R}$ for the transitive and reflexive closure of $\to_\mathcal{R}$. We often write $\to$ instead of $\to_\mathcal{R}$ when $\mathcal{R}$ is clear from the context. Given a set of equations $\mathcal{E}$, u rewrites modulo $\mathcal{E}$ by $\mathcal{R}$ to v, denoted $u \to_{\mathcal{R}/\mathcal{E}} v$, if $u =_\mathcal{E} t[l\sigma]_p$ and $t[r\sigma]_p =_\mathcal{E} v$ for some context t, position $p \in \mathsf{Pos}(t)$, rule $l \to r$ in $\mathcal{R}$, and substitution σ. $\mathcal{R}$ is *$\mathcal{E}$-terminating* if there are no infinite chains $t_1 \to_{\mathcal{R}/\mathcal{E}} t_2 \to_{\mathcal{R}/\mathcal{E}} \ldots$. $\mathcal{R}$ is *$\mathcal{E}$-confluent* iff whenever $t \to_{\mathcal{R}/\mathcal{E}} u$ and $t \to_{\mathcal{R}/\mathcal{E}} v$, there exist u',v' such that $u \to^*_{\mathcal{R}/\mathcal{E}} u'$, $v \to^*_{\mathcal{R}/\mathcal{E}} v'$, and $u' =_\mathcal{E} v'$. $\mathcal{R}$ is *$\mathcal{E}$-convergent* if it is $\mathcal{E}$-terminating and $\mathcal{E}$-confluent. A term t is in *normal form* with respect to $(\mathcal{R}/\mathcal{E})$ if there is no term s such that $t \to_{\mathcal{R}/\mathcal{E}} s$. If $t \to^*_{\mathcal{R}/\mathcal{E}} s$ and s is in normal form then we say that s is a normal form of t. When this normal form is unique (up to $\mathcal{E}$) we write $s = t\downarrow_{\mathcal{R}/\mathcal{E}}$.

2.3 Frames, deduction and static equivalence

We can assemble messages into a *frame*. A frame $\varphi = \nu\tilde{n}.\sigma$ consists of a finite set of restricted names $\tilde{n}$ and a substitution σ. The domain of a frame φ, denoted $\mathsf{dom}(\varphi)$ is the domain of the underlying substitution, *i.e.*, $\mathsf{dom}(\sigma)$. Given two frames φ_1 and φ_2 we write $\varphi_1 =_\alpha \varphi_2$ when φ_1 and φ_2 are equal up to α-conversion of restricted names.

Given a frame we can define different notions of knowledge of an adversary: deduction and static equivalence. Deduction models the fact that the adversary can compute the value of a given message.

Definition 2.1 *Let $\varphi = \nu\tilde{n}.\sigma$ be a frame. We say that a term t is* deducible *from φ in the equational theory $\mathcal{E}$, written $\varphi \vdash_\mathcal{E} t$ iff there exists a term R such that $\mathit{names}(R) \cap \tilde{n} = \emptyset$ and $R\sigma =_\mathcal{E} t$.*

We call the term R in the above definition a *recipe* and when t can be deduced using the recipe R we sometimes write $\varphi \vdash^R_\mathcal{E} t$.

Another notion of adversary knowledge can be expressed in terms of indistinguishability of frames, called *static equivalence*.

Definition 2.2 *Let φ be a frame. We say that two terms M, N are equal in φ under the equational theory $\mathcal{E}$, written $(M =_{\mathcal{E}} N)\varphi$, iff there exist $\tilde{n}, \sigma$ such that $\varphi =_\alpha \nu\tilde{n}.\sigma$, $fn(M,N) \cap \tilde{n} = \emptyset$ and $M\sigma =_{\mathcal{E}} N\sigma$.*

Two frames φ_1, φ_2 are statically equivalent *in the equational theory $\mathcal{E}$, written $\varphi_1 \sim_{\mathcal{E}} \varphi_2$, iff $\mathsf{dom}(\varphi_1) = \mathsf{dom}(\varphi_2)$ and for all terms M, N $(M =_{\mathcal{E}} N)\varphi_1 \Leftrightarrow (M =_{\mathcal{E}} N)\varphi_2$.*

2.4 The applied pi calculus: syntax and semantics

The applied pi caculus [AF01] is an extension of the pi calculus which allows to transmit complex data terms on channels rather than just names. We suppose that terms are generated by a term algebra as introduced above and respect a simple sort system. We distinguish terms of channel sort and terms of base sort. Function symbols only take arguments of base sort and return results of base sort.

Syntax. In the applied pi calculus, one has *plain processes* and *extended processes*. Plain processes are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than just names). Below, M and N are terms, n is a name, x a variable and u is a metavariable, standing either for a name or a variable. Extended processes add *active substitutions* and restriction on variables.

$P, Q, R :=$ plain processes	$A, B, C :=$ extended processes
0	P
$P \mid Q$	$A \mid B$
$\nu n.P$	$\nu n.A$
if $M = N$ then P else Q	$\nu x.A$
$\mathsf{in}(u, x).P$	$\{^M/_x\}$
$\mathsf{out}(u, N).P$	

$\{^M/_x\}$ is the substitution that replaces the variable x with the term M. Active substitutions generalise "let". The process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to the process "let $x = M$ in P". As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables and free and bound names of A, respectively. We also assume that, in an extended process, there is at most one substitution for each variable, and there is exactly one when the variable is restricted. We say that an extended process is *closed* if all its variables are either bound or defined by an active substitution.

Active substitutions are useful because they allow us to map an extended process A to its *frame* $\phi(A)$ by replacing every plain process in A with 0. (As we will see below any frame is structurally equivalent to a frame as introduced in the previous section.) A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behaviour. The domain of a frame are the variables defined by active substitutions that are not restricted. The domain of a process is the domain of its frame, *i.e.* $\mathsf{dom}(A) = \mathsf{dom}(\phi(A))$. We assume that all variables in the domain of a process are of base sort.

As usual a context is a process with a hole. An evaluation context is a context whose hole is not under an input, output or conditional (equivalently, it is an extended process with a hole instead of an extended process).

Semantics. The semantics of the applied pi calculus is defined by two relations.

Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, application of evaluation contexts, and such that:

Par-0	$A \mid 0$	$\equiv$	A	
Par-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$	
Par-C	$A \mid B$	$\equiv$	$B \mid A$	
New-0	$\nu n.0$	$\equiv$	0	
New-C	$\nu u.\nu v.A$	$\equiv$	$\nu v.\nu u.A$	
New-Par	$A \mid \nu u.B$	$\equiv$	$\nu u.(A \mid B)$	if $u \notin fn(A) \cup fv(A)$
Alias	$\nu x.\{^M/_x\}$	$\equiv$	0	
Subst	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A\{^M/_x\}$	
Rewrite	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$	if $M =_{\mathcal{E}} N$

Internal reduction, noted $\rightarrow$, is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

Comm	$\mathsf{out}(a, M).P \mid \mathsf{in}(a, x).Q$	$\rightarrow$	$P \mid Q\{^M/_x\}$	
Then	if $M = N$ then P else Q	$\rightarrow$	P	where $M =_{\mathcal{E}} N$
Else	if $M = N$ then P else Q	$\rightarrow$	Q	
	for any ground terms M and N such that $M \neq_{\mathcal{E}} N$			

Security properties are generally defined to be closed under application of evaluation contexts, *i.e.*, they must hold when run in parallel with any adversary which can be written as an applied pi calculus process. While this yields intuitive definitions the universal quantification over all evaluation contexts makes proofs difficult. Therefore, Abadi and Fournet [AF01] introduced a labelled semantics which allows processes to directly interact with the environment.

The *labelled semantics* extends the semantics given above by the relation $\xrightarrow{\alpha}$ where α is

- either an input $\mathsf{in}(a, M)$ where a is a channel name and M is a term that can contain names and variables,
- or an output $\mathsf{out}(a, u)$ or $\nu u.\mathsf{out}(a, u)$ where a is a channel name and u either a variable of base type or a channel name.

$$\textsc{In}\quad \mathsf{in}(a,x).P \xrightarrow{in(a,M)} P\{^{M}/_{x}\} \qquad \textsc{Out-Atom}\quad \mathsf{out}(a,u).P \xrightarrow{out(a,u)} P$$

$$\textsc{Open-Atom}\quad \frac{A \xrightarrow{out(a,u)} A' \qquad u \neq a}{\nu u.A \xrightarrow{\nu u.out(a,u)} A'} \qquad \textsc{Scope}\quad \frac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\textsc{Par}\quad \frac{bn(\alpha) \cap fn(B) = \emptyset \quad A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B} \qquad \textsc{Struct}\quad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$$

In the following we decorate relations with * to denote their reflexive, transitive closure and write $\Longrightarrow$ for $((\rightarrow^*) \xrightarrow{\alpha} (\rightarrow^*))^*$, *i.e.* any sequence of internal and labelled reductions.

2.5 Security properties

We now illustrate on some examples how we can express security properties in the applied pi calculus.

Secrecy. Secrecy or confidentiality is one of the most basic properties. As usual in symbolic models we express secrecy of a term as the adversary's inability to interact with the process and reach a state where this term can be deduced. As a first approximation we could state secrecy as follows:

> Let A be a closed extended process and t a closed term. t is secret in A iff for any evaluation context $C[\,]$, we have that if $C[A] \rightarrow^* B$ then $\phi(B) \not\vdash_{\mathcal{E}} t$.

This definition is however not satisfactory: because of α-conversion any term t can always be deduced. Correctly expressing the secrecy of a term is slightly more tricky. One way of doing so (which we adopted in [22]) is by annotating processes with (parametrized) *events*. We will not give a precise formalization of events here and refer the reader to [22] or [Bla09]. We suppose that we annotate a process with an event $\mathsf{secret}(t)$ in order to specify secrecy of the term t. When an event is executed it is stored and we say that the event *holds* from that point on. The events are stored such that they remain in the scope of the restricted names occuring in them so that they are subject to α-conversion.

> Let A be a closed extended process and t a closed term. t is secret in A iff for any evaluation context $C[\,]$, we have that if $C[A] \rightarrow^* B$ then $\mathsf{secret}(t)$ holds in B and $\phi(B) \not\vdash_{\mathcal{E}} t$.

An alternative modelling based on the labelled semantics would be as follows.

> Let A be a closed extended process and t a closed term. t is secret in A iff if $C[A] \Longrightarrow B$ then $\mathsf{secret}(t)$ holds in B and $\phi(B) \not\vdash_{\mathcal{E}} t$.

Correspondence properties. Correspondence assertions again suppose that processes have been annotated with events. A correspondence assertion $\mathsf{end}(\tilde{M}) \Rightarrow \mathsf{begin}(\tilde{N})$ ensures that for any occurrence of the end event there exists a previous occurrence of the corresponding begin event (such that common variables in $\tilde{M}$ and $\tilde{N}$ have been instantiated in the same

way). The correspondence is said to be *injective* if there exists a 1 to 1 mapping between occurrences of the **end** and the **begin** events.

Correspondence assertions have been classically used to model authentication, going back to the work of Woo and Lam [WL92]. We will use such assertions to model a general security property of the TPM in Chapter 4.

Equivalence properties. An important part of our work is related to equivalence properties which can be modelled using observational equivalence. As we will see equivalence properties are particularly useful for modelling anonymity properties, real-or-random properties and indistinguishability from ideal functionalities. We are now defining observational equivalence and observational preorder. For this we introduce the notion of a *barb*. Given an extended process A and a channel name a, we write $A \Downarrow a$ when $A \rightarrow^* C[\mathsf{out}(a, M).P]$ for some term M, plain process P, and evaluation context $C[\_]$ that does not bind a.

Definition 2.3 (observational preorder, equivalence) Observational preorder *($\preceq$) (resp.* equivalence *($\approx$)) is the largest (resp. largest symmetric) relation on extended processes having same domain such that $A \mathcal{R} B$ implies*

1. *if $A \Downarrow a$ then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B';*
3. *$C[A] \mathcal{R} C[B]$ for all closing evaluation contexts $C[\_]$.*

As we have already explained the universal quantification over all evaluation contexts makes proofs complicated. Therefore a more convenient relation to manipulate is (bi)simulation which relies on the labelled semantics.

Definition 2.4 (labelled (bi)similarity) *A relation $\mathcal{R}$ on closed extended processes is a simulation relation if $A \mathcal{R} B$ implies*

1. *$\phi(A) \sim \phi(B)$,*
2. *if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq \mathsf{dom}(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B'.*

If $\mathcal{R}$ and $\mathcal{R}^{-1}$ are both simulation relations we say that $\mathcal{R}$ is a bisimulation relation. Labelled similarity *($\preceq_\ell$), resp.* labelled bisimilarity *($\approx_\ell$), is the largest simulation, resp. bisimulation relation.*

Abadi and Fournet [AF01, Theorem 1] have shown that $\approx = \approx_\ell$. Observational preorder and similarity were not introduced in [AF01]. However, these definitions seem natural and were convenient for our framework of simulation based security (Chapter 6). Obviously we have that $\approx \subset \preceq$ and $\approx_\ell \subset \preceq_\ell$. In [24] we prove that labelled similarity is a precongruence.

Proposition 2.1 *Let A and B be two extended processes such that $A \preceq_\ell B$. We have that $C[A] \preceq_\ell C[B]$ for all closing evaluation context $C[\_]$.*

From this proposition it follows that $\preceq_\ell \subseteq \preceq$. Hence, we can use labelled similarity as a convenient proof technique for observational preorder. We actually expect the two relations to coincide but did not prove it as we did not need it.

CHAPTER 3

Modelling electronic voting protocols and their properties

In this chapter we give an overview of our work on modelling and verifying e-voting protocols. We started this line of work in [37] with a case study of the protocol by Fujioka *et al.* [FOO92]. At that time we identified several challenges that arised when analysing e-voting protocols: such protocols use non-standard cryptographic (*e.g.* blind signatures, trapdoor commitments, non-interactive zero knowledge proofs, ...), the required properties are non trivial to model and the protocols themselves are complex (several authorities, need for synchronisation points, ...) The applied pi calculus seemed to be a good choice as a modelling language as it allows to model cryptographic primitives in a general way by the means of an equational theory, offers a rich language for specifying the protocols and allows to easily specify reachability as well as indistinguishability properties. In our first paper [37] we analysed the protocol by Fujioka *et al.* [FOO92] covering a series of properties, such as fairness (no early results), eligibility (only eligible voter have the right to vote) and anonymity. In the following sections we describe in more details our work on privacy type properties [37, 34, 13, 22] and verifiability properties [20].

3.1 Privacy-type properties

In this section we describe our work on privacy-type properties. We distinguish three such properties:

- Vote-privacy: the fact that a particular voter voted in a particular way is not revealed to anyone.
- Receipt-freeness: a voter does not gain any information (a *receipt*) which can be used to prove to a coercer that she voted in a certain way.
- Coercion-resistance: a voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

The weakest of the three, called *vote-privacy*, roughly states that the fact that a voter voted in a particular way is not revealed to anyone. When stated in this simple way, however, the property is in general false, because if all the voters vote unanimously then everyone will get to know how everyone else voted. The formalisation we give in fact says that no party receives information which would allow them to distinguish one situation from another one in which two voters swap their votes.

Receipt-freeness says that the voter does not obtain any artefact (a "receipt") which can be used later to prove to another party how she voted. Such a receipt may be intentional or unintentional on the part of the designer of the system. Unintentional receipts might include nonces or keys which the voter is given during the protocol. Receipt-freeness is a stronger property than privacy. Intuitively, privacy says that an attacker cannot discern how a voter votes from any information that the voter necessarily reveals during the course of the election. Receipt-freeness says the same thing even if the voter voluntarily reveals additional information.

Coercion-resistance is the third and strongest of the three privacy properties. Again, it says that the link between a voter and her vote cannot be established by an attacker, this time even if the voter cooperates with the attacker during the election process. Such cooperation can include giving to the attacker any data which she gets during the voting process, and using data which the attacker provides in return. When analysing coercion-resistance, we assume that the voter and the attacker can communicate and exchange data at any time during the election process. Coercion-resistance is intuitively stronger than receipt-freeness, since the attacker has more capabilities.

Of course, the voter can simply tell an attacker how she voted, but unless she provides convincing evidence the attacker has no reason to believe her. Receipt-freeness and coercion-resistance assert that she cannot provide convincing evidence. Coercion-resistance cannot possibly hold if the coercer can physically vote on behalf of the voter. Some mechanism is necessary for isolating the voter from the coercer at the moment she casts her vote. This can be realised by a voting booth, which we model here as a private and anonymous channel between the voter and the election administrators.

Note that in literature the distinction between receipt-freeness and coercion-resistance is not very clear. The definitions are usually given in natural language and are insufficiently precise to allow comparison. The notion of receipt-freeness first appeared in the work of Benaloh and Tuinstra [BT94]. Since then, several schemes [BT94, Oka96] were proposed in order to meet the condition of receipt-freeness, but later shown not to satisfy it. One of the reasons for such flaws is that no formal definition of receipt-freeness has been given. The situation for coercion-resistance is similar. Systems have been proposed aiming to satisfy it; for example, Okamoto [Oka97] presents a system resistant to interactive coercers, thus aiming to satisfy what we call coercion-resistance, but this property is stated only in natural language. A rigorous definition in a computational model has been proposed by Juels *et al.* for coercion-resistance [JCJ05] and in the UC framework by Canetti and Gennaro [CG96], Moran and Naor [MN06] and Unruh and Müller-Quade [UM10]. To the best of our knowledge our definition is the first "formal methods" definition of receipt-freeness and coercion-resistance. It is difficult to compare our definition and the ones proposed in [JCJ05, MN06, UM10] due to the inherently different models. Our work has later been extended by Backes *et al.* [BHM08] who aim automation of coercion-resistance

using ProVerif. There exist also recent definitions of receipt-freeness and coercion resistance in epistemic [BRS07, KT09] and game-theoretic [KTV10] models.

This section is based on the results published in [37, 34, 13].

3.1.1 Formalising voting protocols

Before formalising security properties, we need to define what is an electronic voting protocol in the applied pi calculus. Different voting protocols often have substantial differences. However, we believe that a large class of voting protocols can be represented by processes corresponding to the following structure.

Definition 3.1 (Voting process) *A voting process is a closed plain process*

$$VP \equiv \nu\tilde{n}.(V\sigma_1 \mid \cdots \mid V\sigma_n \mid A_1 \mid \cdots \mid A_m).$$

The $V\sigma_i$ are the voter processes, the A_js the election authorities which are required to be honest and the $\tilde{n}$ are channel names. We also suppose that $v \in \mathsf{dom}(\sigma_i)$ is a variable which refers to the value of the vote. We define an evaluation context S which is as VP, but has a hole instead of two of the $V\sigma_i$.

In order to prove a given property, we may require some of the authorities to be honest, while other authorities may be assumed to be corrupted by the attacker. The processes $A_1, \ldots, A_m$ represent the authorities which are required to be honest. The authorities under control of the attacker need not be modelled, since we consider any possible behaviour for the attacker (and therefore any possible behaviour for corrupt authorities). In this case the communication channels are available to the environment.

3.1.2 Vote-privacy

The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. Anonymity and privacy properties have been successfully studied using equivalences, *e.g.* [SS96] . However, the definition of privacy in the context of voting protocols is rather subtle. While generally most security properties should hold against an arbitrary number of dishonest participants, arbitrary coalitions do not make sense here. Consider for instance the case where all but one voter are dishonest: as the results of the vote are published at the end, the dishonest voter can collude and determine the vote of the honest voter. A classical trick for modelling anonymity is to ask whether two processes, one in which V_A votes and one in which V_B votes, are equivalent. However, such an equivalence does not hold here as the voters' identities are revealed (and they need to be revealed at least to the administrator to verify eligibility). In a similar way, an equivalence of two processes where only the vote is changed does not hold, because the votes are published at the end of the protocol. To ensure privacy we need to hide the *link* between the voter and the vote and not the voter or the vote itself.

In order to give a reasonable definition of privacy, we need to suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a, respectively b. We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is observationally equivalent to a process where V_A votes b and V_B votes a. Formally, privacy is defined as follows.

Definition 3.2 (Vote-privacy) *A voting protocol respects* vote-privacy *(or just* privacy*) if*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx_\ell S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

for all possible votes a and b.

The intuition is that if an intruder cannot detect if arbitrary honest voters V_A and V_B swap their votes, then in general he cannot know anything about how V_A (or V_B) voted. Note that this definition is robust even in situations where the result of the election is such that the votes of V_A and V_B are necessarily revealed. For example, if the vote is unanimous, or if all other voters reveal how they voted and thus allow the votes of V_A and V_B to be deduced.

As already noted, in some protocols the vote-privacy property may hold even if authorities are corrupt, while other protocols may require the authorities to be honest. When proving privacy, we choose which authorities we want to model as honest, by including them in Definition 3.1 of VP (and hence S).

3.1.3 Receipt-Freeness

Similarly to privacy, receipt-freeness may be formalised as an observational equivalence. We also formalise receipt-freeness using observational equivalence. However, we need to model the fact that V_A is willing to provide secret information, i.e., the receipt, to the coercer. We assume that the coercer is in fact the attacker who, as usual in the Dolev-Yao model, controls the public channels. To model V_A's communication with the coercer, we consider that V_A executes a voting process which has been modified. We denote by P^{ch} the plain process P that is modified as follows: any input of base type and any freshly generated names of base type are output on channel ch. We do not forward restricted channel names, as these are used for modelling purposes, such as physically secure channels, *e.g.* the voting booth, or the existence of a PKI which securely distributes keys (the keys themselves are forwarded but not the secret channel name on which the keys are received). In the remainder, we assume that $ch \notin fn(P) \cup bn(P)$ before applying the transformation. Given an extended process A and a channel name ch, we to define the extended process $A^{\backslash out(ch,\cdot)}$ as $\nu ch.(A \mid !\mathsf{in}(ch, x))$. Intuitively, such a process is as the process A, but hiding the outputs on the channel ch.

We are now ready to define receipt-freeness. Intuitively, a protocol is receipt-free if, for all voters V_A, the process in which V_A votes according to the intruder's wishes is indistinguishable from the one in which she votes something else. As in the case of privacy, we express this as an observational equivalence to a process in which V_A swaps her vote with V_B, in order to avoid the case in which the intruder can distinguish the situations merely by counting the votes at the end. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows.

Definition 3.3 (Receipt-freeness) *A voting protocol is* receipt-free *if there exists a closed plain process V' such that*

- $V'^{\backslash out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,

- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]$,

for all possible votes a and c.

As before, the context S in the second equivalence includes those authorities that are assumed to be honest. V' is a process in which voter V_A votes a but communicates with the coercer C in order to feign cooperation with him. Thus, the second equivalence says that the coercer cannot tell the difference between a situation in which V_A genuinely cooperates with him in order to cast the vote c and one in which she pretends to cooperate but actually casts the vote a, provided there is some counter-balancing voter that votes the other way around. The first equivalence of the definition says that if one ignores the outputs V' makes on the coercer channel *chc*, then V' looks like a voter process V_A voting a.

The first equivalence of the definition may be considered too strong; informally, one might consider that the equivalence should be required only in a particular S context rather than requiring it in any context (with access to all the private channels of the protocol). This would result in a weaker definition, although one which is more difficult to work with. In fact, the variant definition would be only slightly weaker; it is hard to construct a natural example which distinguishes the two possibilities, and in particular it makes no difference to the case studies of later sections. Therefore, we prefer to stick to Definition 3.3.

According to intuition, if a protocol is receipt-free (for a given set of honest authorities), then it also respects privacy (for the same set):

Proposition 3.1 *If a voting protocol is receipt-free then it also respects privacy.*

3.1.4 Coercion-Resistance

Coercion-resistance is a stronger property as we give the coercer the ability to communicate *interactively* with the voter and not only receive information. In this model, the coercer can prepare the messages he wants the voter to send. As for receipt-freeness, we modify the voter process. In the case of coercion-resistance, we give the coercer the possibility to provide the messages the voter should send. The coercer can also decide how the voter branches on *if*-statements.

We denote by P^{c_1,c_2} the plain process P that is modified as follows: any input of base type and any freshly generated names of base type are output on channel c_1. Moreover, when M is a term of base type, any output $\mathsf{out}(u, M)$ is replaced by $\mathsf{in}(c_2, x).\mathsf{out}(u, x)$ where x is a fresh variable and any occurrence of if $M = N$ is replaced by $\mathsf{in}(c_2, x).$ if $x =$ true.

As a first approximation, we could try to define coercion-resistance in the following way: a protocol is coercion-resistant if there is a V' such that

$$S[V_A\{^?/_v\}^{c_1,c_2} \mid V_B\{^a/_v\}] \approx_\ell S[V' \mid V_B\{^c/_v\}]. \tag{3.1}$$

On the left, we have the coerced voter $V_A\{^?/_v\}^{c_1,c_2}$; no matter what she intends to vote (the "?"), the idea is that the coercer will force her to vote c. On the right, the process V' resists coercion, and manages to vote a. Unfortunately, this characterisation has the problem that the coercer could oblige $V_A\{^?/_v\}^{c_1,c_2}$ to vote $c' \neq c$. In that case, the process

$V_B\{^c/_v\}$ would not counter-balance the outcome to avoid a trivial way of distinguishing the two sides.

To enable us to reason about the coercer's choice of vote, we model the coercer's behaviour as a context C that defines the interface c_1, c_2 for the voting process. The context C coerces a voter to vote c. Thus, we can characterise coercion-resistance as follows: a protocol is coercion-resistant if there is a V' such that

$$S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}], \tag{3.2}$$

where C is a context ensuring that the coerced voter $V_A\{^?/_v\}^{c_1,c_2}$ votes c. The context C models the coercer's behaviour, while the environment models the coercer's powers to observe whether the coerced voter behaves as instructed. We additionally require that the context C does not directly use the channel names $\tilde{n}$ restricted by S. Formally one can ensure that $V_A\{^?/_v\}^{c_1,c_2}$ votes c by requiring that $C[V_A\{^?/_v\}^{c_1,c_2}] \approx_\ell V_A\{^c/_v\}^{chc}$. We actually require a slightly weaker condition, $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$, which results in a stronger property. Backes *et al.* [BHM08] propose a variant of our definitions: instead of forcing the coercer's vote to c, they require the existence of an *extractor* process which extracts the vote of the coercer to enable counter-balancing.

Putting the above ideas together, we get to the following definition:

Definition 3.4 (Coercion-resistance) *A voting protocol is* coercion-resistant *if there exists a closed plain process V' such that for any $C = \nu c_1.\nu c_2.(\_ \mid P)$ satisfying $\tilde{n} \cap fn(C) = \emptyset$ and $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$, we have*

- $C[V']^{\setminus out(chc,\cdot)} \approx_\ell V_A\{^a/_v\}$,
- $S[C[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}] \approx_\ell S[C[V'] \mid V_B\{^c/_v\}]$.

Note that $V_A\{^?/_v\}^{c_1,c_2}$ does not depend on what we put for "?".

The condition that $S[C[V_A\{^?/_v\}^{c1,c2}] \mid V_B\{^a/_v\}] \approx_\ell S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}]$ means that the context C outputs the secrets generated during its computation; this is required so that the environment can make distinctions on the basis of those secrets, as in receipt-freeness. The first bullet point expresses that V' is a voting process for A which fakes the inputs/outputs with C and succeeds in voting a in spite of the coercer. The second bullet point says that the coercer cannot distinguish between V' and the really coerced voter, provided another voter V_B counter-balances.

As in the case of receipt-freeness, the first equivalence of the definition could be made weaker by requiring it only in a particular S context. But we chose not to adopt this extra complication, for the same reasons as given in the case of receipt-freeness.

Remark 3.1 *The context C models the coercer's behaviour; we can see its role in equivalence (3.2) as imposing a restriction on the distinguishing power of the environment in equivalence (3.1). Since the coercer's behaviour is modelled by C while its distinguishing powers are modelled by the environment, it would be useful to write (3.2) as*

$$C[S[V_A\{^?/_v\}^{c_1,c_2}] \mid V_B\{^a/_v\}]] \approx_\ell C[S[V' \mid V_B\{^c/_v\}]]. \tag{3.3}$$

We have shown that equivalences (3.2) and (3.3) are the same.

Remark 3.2 *Note that our definition of coercion-resistance cannot cover attacks such as the* ballot-as-signature attack *(also known as the* Italian attack*) [DC] where the number of possible votes is extremely high and therefore a particular vote is unlikely to appear twice and can hence be identified by a coercer. Such an attack is not in the scope of our definition, as we suppose in our definition that there exists a second voter that votes as the coercer requested.*

However, our definitions can capture attacks such as vote duplication *attacks where the attacker duplicates your vote (without knowing its value) in order to deduce it from the result. Smyth and Cortier [SC10] have demonstrated that such an attack exists on the Helios protocol using our definition of privacy.*

According to intuition, if a protocol is coercion-resistant then it respects receipt-freeness too (as before, we keep constant the set of honest authorities):

Proposition 3.2 *If a voting protocol is coercion-resistant then it also respects receipt-freeness.*

3.1.5 Case studies

We have analysed the above discussed privacy-type properties for three protocols: Fujioka *et al.* [FOO92], Okamoto [Oka96] and Lee *et al.* [LBD+04]. As we only model authorities that are required to be honest for these protocols to hold we are able to identify which authorities need to be trusted for these particular properties. When analysing these three properties the existence of the process V' for receipt-freeness and coercion-resistance turned out to be easy. In the protocol specification these processes are generally described as the way of achieving the properties. The equivalence properties could however not be proved automatically and required hand proofs. We were however able to rely on ProVerif for some Lemmas on static equivalence. We summarise the results of these three case studies in Figure 3.1.

Property	*Fujioka et al.*	*Okamoto*	*Lee et al.*
Vote-privacy trusted authorities	✓ none	✓ timeliness mbr.	✓ administrator
Receipt-freeness trusted authorities	× n/a	✓ timeliness mbr.	✓ admin. & collector
Coercion-resistance trusted authorities	× n/a	× n/a	✓ admin. & collector

Figure 3.1: Summary of protocols and properties

3.2 An epistemic logic for the applied pi calculus

As already argued, the applied pi calculus is a convenient and flexible formalism for describing the processes which model the protocol. However, security properties are more difficult to specify. As we have seen some properties may directly be specified using observational equivalence, but this is generally not very natural and convenient. A sometimes

more natural approach to verify protocols for correctness would be to define a suitable logic interpreted over the terms of the calculus and express the desired security goal in that logic.

In [22] we define an epistemic logic for the applied pi calculus suitable for expressing security goals. The logic itself is an LTL like temporal logic with a special predicate Has that models deducibility of messages by an intruder and a knowledge operator K which allows us to reason about the intruder's knowledge, as defined in epistemic logics. In epistemic reasoning, such as in [FHMV95], an agent *knows* a formula ϕ if ϕ holds on all runs that are "equivalent" to the current run. This notion of equivalent runs models imperfect knowledge of the current state: an observer can only identify a set of possible runs which are compatible with its partial view of the system. In the context of security protocols imperfect information may for instance stem from the fact that a message is encrypted with a secret key. In [22] we define the partial view of the adversary to be the set of all traces that are statically equivalent (naturally lifted to traces) to the actually observed trace. Epistemic logics have already been used to model security properties, *e.g.* in [HO05]. However, these works rarely specify a modelling language for the protocols, starting directly from a set of traces equipped with a partitioning of this set modelling partial views. In our work we provide a concrete modelling language (the applied pi calculus) and an equivalence relation (static equivalence on traces) which allows us to both specify the protocol and the properties.

Our logic can be used for a range of security properties: secrecy, authentication as well as fairness in contract signing protocols. In particular we can specify *privacy* in voting protocols, which relies on the epistemics of the intruder. We show that a definition of vote privacy in terms of process equivalence as defined in the previous section implies vote privacy in terms of epistemic logic, as defined in [BRS07] (and rephrased in our logic). When we slightly weaken the equivalence based definition, replacing observational equivalence with trace equivalence, under reasonable assumptions, we show that the converse implication, i.e. epistemic privacy implies privacy as equivalence, also holds. This result is important in that it clarifies the relationship between two definitions of privacy employed in the literature. Furthermore, the result suggests that trace equivalence might be more appropriate to model vote privacy even though observational equivalence is often more convenient to manipulate.

3.3 Election verifiability

We present a definition of election verifiability which captures three desirable aspects: individual, universal and eligibility verifiability. We formalise verifiability as a triple of Boolean tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}, \Phi^{\mathsf{EV}}$ which are required to satisfy several conditions on all possible executions of the protocol. Φ^{IV} is intended to be checked by the individual voter who instantiates the test with her private information (*e.g.*, her vote and data derived during the execution of the protocol) and the public information available on the bulletin board. Φ^{UV} and Φ^{EV} can be checked by any external observer and only rely on public information, *i.e.*, the contents of the bulletin board.

The consideration of eligibility verifiability is particularly interesting as it provides an assurance that the election outcome corresponds to votes legitimately cast and hence

provides a mechanism to detect ballot stuffing. We note that this property has been largely neglected in previous work and an earlier work of ours [51] only provided limited scope for.

A further interesting aspect of our work is the clear identification of which parts of the voting system need to be trusted to achieve verifiability. All untrusted parts of the system will be controlled by the adversarial environment and do not need to be modelled. Ideally, such a process would only model the interaction between a *voter* and the voting terminal; *that is, the messages input by the voter.* In particular, the voter should not need to trust the election hardware or software. However, achieving absolute verifiability in this context is difficult and one often needs to trust some parts of the voting software or some administrators. Such trust assumptions are motivated by the fact that parts of a protocol can be audited, or can be executed in a distributed manner amongst several different election officials. For instance, in Helios 2.0 [Adi08], the ballot construction can be audited using a cast-or-audit mechanism. Whether trust assumptions are reasonable depends on the context of the given election, but our work makes them explicit.

Of course the tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}$ and Φ^{EV} need to be verified in a trusted environment (if a test is checked by malicious software that always evaluates the test to hold, it is useless). However, the verification of these tests, unlike the election, can be repeated on different machines, using different software, provided by different stakeholders of the election. Another possibility to avoid this issue would be to have tests which are human-verifiable as discussed in [Adi06, Chapter 5].

This section is based on the results presented in [20].

3.3.1 Formalising voting protocols for verifiability properties

To model verifiability properties we add a *record* construct to the applied pi calculus. We assume an infinite set of distinguished *record variables* $r, r_1, \ldots$. The syntax of plain processes is extended by the construct $\mathsf{rec}(r, M).P$. We write $fn(A)$ and $fn(M)$ for the set of record variables in a process and a term. Intuitively, the record message construct $\mathsf{rec}(r, M).P$ introduces the possibility to enter special entries in frames. We suppose that the sort system ensures that r is a variable of record sort, which may only be used as a first argument of the rec construct or in the domain of the frame. Moreover, we make the global assumption that a record variable has a unique occurrence in each process. Intuitively, this construct will be used to allow a voter to privately record some information which she may later use to verify the election.

As discussed in the introduction we want to explicitly specify the parts of the election protocol which need to be trusted. Formally the trusted parts of the voting protocol can be captured using a voting process specification.

Definition 3.5 (Voting process specification) *A* voting process specification *is a tuple* $\langle V, A\rangle$ *where* V *is a plain process without replication and* A *is a closed evaluation context such that* $fv(V) = \{v\}$ *and* $fn(V) = \emptyset$.

For the purposes of individual verifiability the voter may rely on some data derived during the protocol execution. We must therefore keep track of all such values, which is achieved using the record construct. Given a finite process P without replication we denote by $\mathsf{R}(P)$, the process which records any freshly generated name and any input, *i.e.*, we

replace any occurrence of νn with $\nu n.\mathsf{rec}(r, n)$ and $\mathsf{in}(u, x)$ with $\mathsf{in}(u, x).\mathsf{rec}(r, x)$ for some fresh record variable r for each replacement.

Definition 3.6 *Given a voting process specification* $\langle V, A\rangle$*, integer* $n \in \mathbb{N}$*, and names* $s_1, \ldots, s_n$*, we build the augmented voting process* $\mathsf{VP}_n^+(s_1, \ldots, s_n) = A[V_1^+ \mid \cdots \mid V_n^+]$ *where* $V_i^+ = \mathsf{R}(V)\{^{s_i}/_v\}\{^{r_i}/_r \mid r \in fn(\mathsf{R}(V))\}$.

Given a sequence of record variables $\tilde{r}$, we denote by $\tilde{r}_i$ the sequence of variables obtained by indexing each variable in $\tilde{r}$ with i. The process $\mathsf{VP}_n^+(s_1, \ldots, s_n)$ models the voting protocol for n voters casting votes $s_1, \ldots, s_n$, who privately record the data that may be needed for verification using record variables $\tilde{r}_i$.

3.3.2 Election verifiability

We formalise election verifiability using three tests Φ^{IV}, Φ^{UV}, Φ^{EV}. Formally, a test is built from conjunctions and disjunctions of *atomic tests* of the form $(M =_E N)$ where M, N are terms. Tests may contain variables and will need to hold on frames arising from arbitrary protocol executions. We now recall the purpose of each test and assume some naming conventions about variables.

Individual verifiability: The test Φ^{IV} allows a voter to identify her ballot in the bulletin board. The test has:

- a variable v referring to a voter's vote.
- a variable w referring to a voter's public credential.
- some variables $x, \bar{x}, \hat{x}, \ldots$ expected to refer to global public values pertaining to the election, *e.g.*, public keys belonging to election administrators.
- a variable y expected to refer to the voter's ballot on the bulletin board.
- some record variables $r_1, \ldots, r_k$ referring to the voter's private data.

Universal verifiability: The test Φ^{UV} allows an observer to check that the election outcome corresponds to the ballots in the bulletin board. The test has:

- a tuple of variables $\tilde{v} = (v_1, \ldots, v_n)$ referring to the declared outcome.
- some variables $x, \bar{x}, \hat{x}, \ldots$ as above.
- a tuple $\tilde{y} = (y_1, \ldots, y_n)$ expected to refer to all the voters' ballots on the bulletin board.
- some variables $z, \bar{z}, \hat{z}, \ldots$ expected to refer to outputs generated during the protocol used for the purposes of universal and eligibility verification.

Eligibility verifiability: The test Φ^{EV} allows an observer to check that each ballot in the bulletin board was cast by a unique registered voter. The test has:

- a tuple $\tilde{w} = (w_1, \ldots, w_n)$ referring to public credentials of eligible voters.
- a tuple $\tilde{y}$, variables $x, \bar{x}, \hat{x}, \ldots$ and variables $z, \bar{z}, \hat{z}, \ldots$ as above.

3.3.2.1 Individual and universal verifiability.

The tests suitable for the purposes of election verifiability have to satisfy certain conditions: if the tests succeed, then the data output by the election is indeed valid (*soundness*); and there is a behaviour of the election authority which produces election data satisfying the tests (*effectiveness*). Formally these requirements are captured by the definition below. We write $\tilde{T} \simeq \tilde{T}'$ to denote that the tuples $\tilde{T}$ and $\tilde{T}'$ are a permutation of each others modulo the equational theory, that is, we have $\tilde{T} = T_1, \ldots T_n$, $\tilde{T}' = T'_1, \ldots T'_n$ and there exists a permutation χ on $\{1, \ldots, n\}$ such that for all $1 \leq i \leq n$ we have $T_i =_E T'_{\chi(i)}$.

Definition 3.7 (Individual and universal verifiability) *A voting specification $\langle V, A\rangle$ satisfies* individual and universal verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}$ such that $fn(\Phi^{\mathsf{IV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{UV}}) = \emptyset$, $fn(\Phi^{\mathsf{IV}}) \subseteq fn(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ the conditions below hold. Let $\tilde{r} = fn(\Phi^{\mathsf{IV}})$ and $\Phi^{\mathsf{IV}}_i = \Phi^{\mathsf{IV}}\{^{s_i}/_{v}, ^{\tilde{r}_i}/_{\tilde{r}}\}$.*

Soundness. *For all contexts C and processes B such that $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:*

$$\forall i, j. \quad \Phi^{\mathsf{IV}}_i\sigma \wedge \Phi^{\mathsf{IV}}_j\sigma \Rightarrow i = j \tag{3.4}$$

$$\Phi^{\mathsf{UV}}\sigma \wedge \Phi^{\mathsf{UV}}\{^{\tilde{v}'}/_{\tilde{v}}\}\sigma \Rightarrow \tilde{v}\sigma \simeq \tilde{v}'\sigma \tag{3.5}$$

$$\bigwedge_{1\leq i\leq n} \Phi^{\mathsf{IV}}_i\{^{y_i}/_{y}\}\sigma \wedge \Phi^{\mathsf{UV}}\sigma \;\Rightarrow\; \tilde{s} \simeq \tilde{v}\sigma \tag{3.6}$$

Effectiveness. *There exists a context C and a process B, such that $C[\mathsf{VP}^+_n(s_1, \ldots, s_n)] \Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and*

$$\bigwedge_{1\leq i\leq n} \Phi^{\mathsf{IV}}_i\{^{y_i}/_{y}\}\sigma \wedge \Phi^{\mathsf{UV}}\sigma \tag{3.7}$$

An individual voter should verify that the test Φ^{IV} holds when instantiated with her vote s_i, the information $\tilde{r}_i$ recorded during the execution of the protocol and some bulletin board entry. Indeed, Condition (3.4) ensures that the test will hold for at most one bulletin board entry. (Note that Φ^{IV}_i and Φ^{IV}_j are evaluated with the same ballot $y\sigma$ provided by $C[\_]$.) The fact that her ballot is counted will be ensured by Φ^{UV} which should also be tested by the voter. An observer will instantiate the test Φ^{UV} with the bulletin board entries $\tilde{y}$ and the declared outcome $\tilde{v}$. Condition (3.5) ensures the observer that Φ^{UV} only holds for a single outcome. Condition (3.6) ensures that if a bulletin board contains the ballots of voters who voted $s_1, \ldots, s_n$ then Φ^{UV} only holds if the declared outcome is (a permutation of) these votes. Finally, Condition (3.7) ensures that there exists an execution where the tests hold. In particular this allows us to verify whether the protocol can satisfy the tests when executed as expected. This also avoids tests which are always false and would make Conditions (3.4)–(3.6) vacuously hold.

3.3.2.2 Eligibility verifiability.

To fully capture *election verifiability*, the tests Φ^{IV} and Φ^{UV} must be supplemented by a test Φ^{EV} that checks eligibility of the voters whose votes have been counted. We suppose that the public credentials of eligible voters appear on the bulletin board. Φ^{EV} allows an observer to check that only these individuals (that is, those in possession of credentials) cast votes, and at most one vote each.

Definition 3.8 (Election verifiability) *A voting specification $\langle V, A\rangle$ satisfies* election verifiability *if for all $n \in \mathbb{N}$ there exist tests $\Phi^{\mathsf{IV}}, \Phi^{\mathsf{UV}}, \Phi^{\mathsf{EV}}$ such that $fn(\Phi^{\mathsf{IV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{EV}}) = fn(\Phi^{\mathsf{UV}}) = fn(\Phi^{\mathsf{EV}}) = \emptyset$, $fn(\Phi^{\mathsf{IV}}) \subseteq fn(\mathsf{R}(V))$, and for all names $\tilde{s} = (s_1, \ldots, s_n)$ we have:*

1. *The tests Φ^{IV} and Φ^{UV} satisfy each of the conditions of Definition 3.7;*
2. *The additional conditions 3.8, 3.9, 3.10 and 3.11 below hold.*

Let $\tilde{r} = fn(\Phi^{\mathsf{IV}})$, $\Phi_i^{\mathsf{IV}} = \Phi^{\mathsf{IV}}\{{}^{s_i}/_{v}, {}^{\tilde{r}_i}/_{\tilde{r}}, {}^{y_i}/_{y}\}$, $X = fv(\Phi^{\mathsf{EV}}) \backslash \mathsf{domVP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)$

Soundness. *For all contexts C and processes B such that $C[\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)] \Longrightarrow B$ and $\phi(B) \equiv \nu\tilde{n}.\sigma$, we have:*

$$\Phi^{\mathsf{EV}}\sigma \wedge \Phi^{\mathsf{EV}}\{{}^{x'}/_{x} \mid x \in X \backslash \tilde{y}\}\sigma \Rightarrow \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{3.8}$$

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{\mathsf{IV}}\sigma \wedge \Phi^{\mathsf{EV}}\{{}^{\tilde{w}'}/_{\tilde{w}}\}\sigma \quad \Rightarrow \quad \tilde{w}\sigma \simeq \tilde{w}'\sigma \tag{3.9}$$

$$\Phi^{\mathsf{EV}}\sigma \wedge \Phi^{\mathsf{EV}}\{{}^{x'}/_{x} \mid x \in X \backslash \tilde{w}\}\sigma \Rightarrow \tilde{y}\sigma \simeq \tilde{y}'\sigma \tag{3.10}$$

Effectiveness. *There exists a context C and a process B such that $C[\mathsf{VP}_{\mathsf{n}}^{+}(s_1, \ldots, s_n)]$ $\Longrightarrow B$, $\phi(B) \equiv \nu\tilde{n}.\sigma$ and*

$$\bigwedge_{1 \leq i \leq n} \Phi_i^{\mathsf{IV}}\sigma \wedge \Phi^{\mathsf{UV}}\sigma \wedge \Phi^{\mathsf{EV}}\sigma \tag{3.11}$$

The test Φ^{EV} is instantiated by an observer with the bulletin board. Condition (3.8) ensures that, given a set of ballots $\tilde{y}\sigma$, provided by the environment, Φ^{EV} succeeds only for one list of voter public credentials. Condition (3.9) ensures that if a bulletin board contains the ballots of voters with public credentials $\tilde{w}\sigma$ then Φ^{EV} only holds on a permutation of these credentials. Condition (3.10) ensures that, given a set of credentials $\tilde{w}$, only one set of bulletin board entries $\tilde{y}$ are accepted by Φ^{EV} (observe that for such a strong requirement to hold we expect the voting specification's frame to contain a public key, to root trust). Finally, the effectiveness condition is similar to Condition (3.7) of Definition 3.7.

3.3.3 Case studies

We have analysed verifiability in three protocols: the Fujioka *et al.* [FOO92], Helios 2.0 [AdMPQ09] and JCJ [JCJ05] recently implemented as Civitas [CCM08]. In particular for each of these protocols we identify the exact parts of the system and software that need to be trusted. As an illustration we consider the protocol by Fujioka *et al.* [FOO92].

Definition 3.9 *The voting process specification* $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}} \rangle$ *is defined as*

$$V_{\mathsf{foo}} \mathrel{\hat{=}} \nu rnd.\mathsf{out}v.\mathsf{out}rnd \quad \textit{and} \quad A_{\mathsf{foo}}[\_\,] \mathrel{\hat{=}} \_$$

Intuitively, this specification says that the voter only needs to enter into a terminal a fresh random value rnd and a vote v. The voter does not need to trust any other parts of the system or the administrators. Whether, a voter can generate a fresh random value (which is expected to be used as the key for a commitment), enter it in a terminal and remember it for verifiability or whether some software is trustworthy to achieve this task is questionable. Our analysis makes this hypothesis explicit. We have shown that the above voting specification indeed respects individual and universal verifiability.

Theorem 3.1 $\langle V_{\mathsf{foo}}, A_{\mathsf{foo}} \rangle$ *satisfies individual and universal verifiability.*

However, the protocol by Fujioka *et al.* does not satisfy eligibility verifiability (even if all the parts of the protocol are trusted).

Similarly, Helios 2.0 does only satisfy individual and universal verifiability. We assume the following parts to be trusted.

- The browser script that constructs the ballot. Although the voter cannot verify it, the trust in this script is motivated by the fact that she is able to audit it. She does that by creating as many ballots as she likes and checking all but one of them, and then casting the one she didn't verify.

- The trustees. Although the trustees' behaviour cannot be verified, voters and observers may want to trust them because trust is distributed among them.

JCJ/Civitas does achieve full election verifiability considering the following trust assumptions.

- The voter is able to construct her ballot; that is, she is able to generate nonces m, m', construct a pair of ciphertexts and generate a zero-knowledge proof.

- The registrar constructs distinct credentials d for each voter and constructs the voter's public credential correctly. (The latter assumption can be dropped if the registrar provides a proof that the public credential is correctly formed [JCJ05].) The registrar also keeps the private part of the signing key secret.

The analyses were carried out by hand, but the proofs were surprisingly straightforward.

3.4 Conclusion and perspectives

In this chapter we summarised our work on modelling and analysis of electronic voting properties. We have provided first symbolic definitions for privacy type properties such as receipt-freeness and coercion-resistance as well as verifiability properties. In particular we identified eligibility verifiability which is a crucial aspect as it allows to verify that no "ballot stuffing" occurred and which has generally been neglected. It would be interesting to analyse more protocols and compare them on the basis of the parts that need to be trusted in order to achieve verifiability. We would also like to investigate whether the notion of receipt-freeness/coercion-resistance and verifiability could be useful in other applications where anonymity would need to be enforced or results to be verified. For example, Dong *et al.* [DJP10] have used our definitions to study receipt-freeness in an auction protocol.

We are also planning to extend our epistemic logic. In particular we plan to introduce the notion of an agent which would allow us to reason about the knowledge of protocol participants rather than only the attacker. We expect to be able to express Küsters *et al.*'s epistemic definition of coercion-resistance [KT09] in this logic and maybe compare it to our definition. Currently most of our proofs were only possible by hand. Therefore, important future work is to design algorithms and procedures for automatically verifying equivalence properties. This will be the topic of the following chapter. Related to this, we would also like to investigate the decidability of (fragments of) our epistemic logic.

CHAPTER 4

Security APIs

Specialized hardware with tamper resistant memory has been designed to allow the execution of untrusted code without compromising the security of the keys. This kind of hardware ranges from expensive Hardware Security Modules (HSMs) used in banking networks to rather cheap USB tokens implementing PKCS#11 or Trusted Platform Modules (TPMs), included in most modern laptops. The main role of these devices is to offer secure key management (generation, import and export of keys) and usage of these keys for encryption, decryption or signing. The functionalities of these devices are available through an Application Programming Interface (API).

The aim of *API level attacks* is to break a security property by a sequence of calls to this API, in a way not foreseen by the designers. An API can be seen as a collection of short security protocols and analysed as such. The first attacks of this style we are aware of go back to Longley and Rigby [LR92] and Bond and Anderson discovered many more [Bon01, BA01], in particular on the IBM 4758 CCA and the Visa Security Module, two HSMs for securing bank transactions. Many of these attacks exploit the algebraic properties of exclusive or which is extensively used in these devices. Since then, efforts have been made to formally analyse APIs using model checkers, theorem provers, and customised decision procedures [CDS07, CKS07, CM06, Ste05]. However, none of these models account for mutable global state, which was identified by Herzog [Her06] as a major barrier to the application of security protocol analysis tools to verify APIs. In security protocol analysis, it is standard to assume that independent runs or sessions of the protocol share no mutable state. They might share long term keys, and they might have local mutable state (they might create session keys, for example), but independent sessions are assumed not to affect each other. Many security APIs however do not have this characteristic. Indeed, the fact that calls to some functions, for example key import commands, do indeed change the internal state of the device, is vital for the way they function. In this chapter we report on our work on analyzing PKCS#11 [10, 28] and the TPM [19] for whose analyses we needed to take state into account.

4.1 Analysis of PKCS#11

RSA Laboratories Public Key Standards (PKCS) #11 defines the 'Cryptoki' API, designed to be an interface between applications and cryptographic devices such as smartcards, Hardware Security Modules (HSMs), and USB key tokens. It has been widely adopted in industry, promoting interoperability of devices. However, the API as defined in the standard gives rise to a number of serious security vulnerabilities [Clu03]. In practice, vendors try to protect against these by restricting the functionality of the interface, or by adding extra features, the details of which are often hard to determine. This has lead to an unsatisfactory situation in which widely deployed security solutions are using an interface which is known to be insecure if implemented naïvely, and for which there are no well established fixes. The situation is complicated by the variety of scenarios in which PKCS#11 is used: an effective security patch for one scenario may disable functionality that is vital for another.

In this section, we describe our work [28, 10] on analyzing PKCS#11 which aims to lay the foundations for an improvement in this situation by defining a formal model for the operation of PKCS#11 key management commands, proving the decidability of certain security properties in this model, and describing an automated framework for proving these properties for different configurations of the API.

4.1.1 Formal model

To model PKCS#11 and attacker capabilities we define a rule-based description language. It is close to a guarded command language *à la Dijkstra* (see [Dij75]) and to the multi-set rewriting framework for protocol analysis (*e.g.*, [Mit02]). One particular point is that it makes a clean separation between the intruder knowledge part, *i.e.*, the monotonic part, and the current system state which is formalized by the attributes that may be set or unset. The semantics will be defined in a classical way as a transition system. We will only give an informal description of the semantics here which will be sufficient to present our results.

Modelling messages and attributes. As usual we suppose that messages are modelled by a term algebra. We will denote by $\mathcal{PT}(\mathcal{F},\mathcal{N},\mathcal{X})$ the set of *plain terms* modelling messages. We also consider a finite set $\mathcal{A}$ of unary function symbols, disjoint from $\mathcal{F}$ which we call *attributes*. The set of *attribute terms* is defined as

$$\mathcal{AT}(\mathcal{A},\mathcal{F},\mathcal{N},\mathcal{X}) = \{att(t) \mid att \in \mathcal{A}, t \in \mathcal{PT}(\mathcal{F},\mathcal{N},\mathcal{X})\}.$$

We define the set of *terms* as $\mathcal{T}(\mathcal{A},\mathcal{F},\mathcal{N},\mathcal{X}) = \mathcal{AT}(\mathcal{A},\mathcal{F},\mathcal{N},\mathcal{X}) \cup \mathcal{PT}(\mathcal{F},\mathcal{N},\mathcal{X})$. Attribute terms will be interpreted as propositions. A *literal* is of the form a or $\neg a$ where $a \in \mathcal{AT}(\mathcal{A},\mathcal{F},\mathcal{N},\mathcal{X})$.

Modelling language. The description of a system is given as a finite set of rules of the form

$$T; L \xrightarrow{\mathsf{new}\,\tilde{n}} T'; L'$$

$$
\begin{array}{lrcl}
\mathsf{Wrap\,(sym/sym)}: & \mathsf{h}(x_1,y_1),\mathsf{h}(x_2,y_2);\ \mathsf{wrap}(x_1),\mathsf{extract}(x_2) & \to & \mathsf{senc}(y_2,y_1)\\
\mathsf{Wrap\,(sym/asym)}: & \mathsf{h}(x_1,\mathsf{priv}(z)),\mathsf{h}(x_2,y_2);\ \mathsf{wrap}(x_1),\mathsf{extract}(x_2) & \to & \mathsf{aenc}(y_2,\mathsf{pub}(z))\\
\mathsf{Wrap\,(asym/sym)}: & \mathsf{h}(x_1,y_1),\mathsf{h}(x_2,\mathsf{priv}(z));\ \mathsf{wrap}(x_1),\mathsf{extract}(x_2) & \to & \mathsf{senc}(\mathsf{priv}(z),y_1)
\end{array}
$$

$$
\begin{array}{l}
\mathsf{Unwrap\,(sym/sym)}:\\
\qquad \mathsf{h}(x_1,y_2),\mathsf{senc}(y_1,y_2);\ \mathsf{unwrap}(x_1) \xrightarrow{\text{new } n_1} \mathsf{h}(n_1,y_1);\ \mathsf{extract}(n_1),\ \mathsf{L}\\
\mathsf{Unwrap\,(sym/asym)}:\\
\quad \mathsf{h}(x_1,\mathsf{priv}(z)),\mathsf{aenc}(y_1,\mathsf{pub}(z));\ \mathsf{unwrap}(x_1) \xrightarrow{\text{new } n_1} \mathsf{h}(n_1,y_1);\ \mathsf{extract}(n_1),\ \mathsf{L}\\
\mathsf{Unwrap\,(asym/sym)}:\\
\qquad \mathsf{h}(x_1,y_2),\mathsf{senc}(\mathsf{priv}(z),y_2);\ \mathsf{unwrap}(x_1) \xrightarrow{\text{new } n_1} \mathsf{h}(n_1,\mathsf{priv}(z));\ \mathsf{extract}(n_1),\ \mathsf{L}
\end{array}
$$

$$
\begin{array}{ll}
\mathsf{KeyGenerate}: & \xrightarrow{\text{new } n_1,k_1} \mathsf{h}(n_1,k_1);\neg\mathsf{extract}(n_1),\ \mathsf{L}\\
\mathsf{KeyPairGenerate}: & \xrightarrow{\text{new } n_1,s} \mathsf{h}(n_1,s),\mathsf{pub}(s);\ \neg\mathsf{extract}(n_1),\ \mathsf{L}
\end{array}
$$

$$
\begin{array}{lrcl}
\mathsf{SEncrypt}: & \mathsf{h}(x_1,y_1),y_2;\ \mathsf{encrypt}(x_1) & \to & \mathsf{senc}(y_2,y_1)\\
\mathsf{SDecrypt}: & \mathsf{h}(x_1,y_1),\mathsf{senc}(y_2,y_1);\ \mathsf{decrypt}(x_1) & \to & y_2
\end{array}
$$

$$
\begin{array}{lrcl}
\mathsf{AEncrypt}: & \mathsf{h}(x_1,\mathsf{priv}(z)),y_1;\ \mathsf{encrypt}(x_1) & \to & \mathsf{aenc}(y_1,\mathsf{pub}(z))\\
\mathsf{ADecrypt}: & \mathsf{h}(x_1,\mathsf{priv}(z)),\mathsf{aenc}(y_2,\mathsf{pub}(z));\ \mathsf{decrypt}(x_1) & \to & y_2
\end{array}
$$

$$
\begin{array}{lrcl}
\mathsf{Set\_Wrap}: & \mathsf{h}(x_1,y_1);\ \neg\mathsf{wrap}(x_1) & \to & \mathsf{wrap}(x_1)\\
\mathsf{Set\_Encrypt}: & \mathsf{h}(x_1,y_1);\ \neg\mathsf{encrypt}(x_1) & \to & \mathsf{encrypt}(x_1)\\
\vdots & & \vdots & \\
\mathsf{UnSet\_Wrap}: & \mathsf{h}(x_1,y_1);\ \mathsf{wrap}(x_1) & \to & \neg\mathsf{wrap}(x_1)\\
\mathsf{UnSet\_Encrypt}: & \mathsf{h}(x_1,y_1);\ \mathsf{encrypt}(x_1) & \to & \neg\mathsf{encrypt}(x_1)\\
\vdots & & \vdots &
\end{array}
$$

where $\mathsf{L} = \neg\mathsf{wrap}(n_1),\neg\mathsf{unwrap}(n_1),\neg\mathsf{encrypt}(n_1),\neg\mathsf{decrypt}(n_1),\neg\mathsf{sensitive}(n_1)$. The ellipsis in the set and unset rules indicates that similar rules exist for some other attributes.

Figure 4.1: PKCS#11 key management subset.

where T and T' are sets of plain terms in $\mathcal{PT}(\mathcal{F},\mathcal{N},\mathcal{X})$, L and L' are sets of literals and $\tilde{n}$ is a set of names in $\mathcal{N}$. The intuitive meaning of such a rule is the following. The rule can be fired if all terms in T are in the intruder knowledge and if all the literals in L are evaluated to true in the current state. The effect of the rule is that terms in T' are added to the intruder knowledge and the valuation of the attributes is updated to satisfy L'. The $\mathsf{new}\,\tilde{n}$ means that all the names in $\tilde{n}$ need to be replaced by fresh names in T' and L'. This allows us to model nonce or key generation: if the rule is executed several times, the effects are different as different names will be used each time.

We always suppose that L' is satisfiable, *i.e.*, it does not contain both a and $\neg a$. Moreover, we require that $\mathit{names}(T \cup L) = \emptyset$ and $\mathit{names}(T' \cup L') \subseteq \tilde{n}$. We also suppose that any variable appearing in T' also appears in T, *i.e.*, $\mathit{vars}(T') \subseteq \mathit{vars}(T)$, and any variable appearing in L' also appears in L, *i.e.*, $\mathit{vars}(L') \subseteq \mathit{vars}(L)$. These conditions were easily verified in all of our experiments with PKCS#11.

Example 4.1 *As an example consider the rules given in Figure 4.1. They model a part of PKCS#11. We detail the first rule which allows wrapping of a symmetric key with a symmetric key. Let us first explain the modelling of handles:* h *is a symbol of arity 2 which*

allows us to model handles to keys. We will use it with a nonce (formally a name) as the first argument and a key as the second argument. Adding a nonce to the arguments of h *allows us to model several distinct handles to the same key. For instance having two handles* $\mathsf{h}(\mathsf{n}_1, \mathsf{k}_1)$ *and* $\mathsf{h}(\mathsf{n}_2, \mathsf{k}_1)$ *models the fact that two distinct memory locations* n_1 *and* n_2 *hold the same key* k_1.

We can now explain the first rule. Intuitively the rule can be read as follows: if the attacker knows the handle $\mathsf{h}(\mathsf{x}_1, \mathsf{y}_1)$*, a reference to a symmetric key* y_1*, and a second handle* $\mathsf{h}(\mathsf{x}_2, \mathsf{y}_2)$*, a reference to a symmetric key* y_2*, and if the attribute* wrap *is set for the handle* $\mathsf{h}(\mathsf{x}_1, \mathsf{y}_1)$ *(note that the handle is uniquely identified by the nonce* x_1*) and the attribute* $\mathsf{extract}$ *is set for the handle* $\mathsf{h}(\mathsf{x}_2, \mathsf{y}_2)$ *then the attacker may learn the wrapping* $\mathsf{senc}(\mathsf{y}_2, \mathsf{y}_1)$*, i.e., the encryption of* y_2 *with* y_1.

Given a set of such rules $\mathcal{R}$ we define a transition system $(Q, q_0, \rightsquigarrow)$ as follows:

- Q is the set of *states*: each state is a pair (S, V), such that $S \subseteq \mathcal{PT}(\mathcal{F}, \mathcal{N})$ and V is a partial valuation of $\mathcal{AT}(\mathcal{A}, \mathcal{F}, \mathcal{N})$. ($V$ is extended from propositions to literals in the expected way and sets of literals as $V(L) = \bigwedge_{\ell \in L} V(\ell)$.)
- $q_0 = (S_0, V_0)$ is the *initial state*. S_0 is the initial attacker knowledge and V_0 defines the initial valuation of some attributes.
- $\rightsquigarrow \subseteq Q \times Q$ is the transition relation induced by the $\mathcal{R}$ (as explained above).

Queries. Security properties are expressed by the means of *queries*.

Definition 4.1 *A* query *is a pair* (T, L) *where* T *is a set of terms and* L *a set of literals (both are not necessarily ground).*

Intuitively, a query (T, L) is satisfied if there exists a substitution θ such that we can reach a state where the adversary knows all terms in $T\theta$ and all literals in $L\theta$ are evaluated to true.

Definition 4.2 *A transition system* $(Q, q_0, \rightsquigarrow)$ *satisfies a query* (T, L) *iff there exists a substitution* θ *and a state* $(S, V) \in Q$ *such that* $q_0 \rightsquigarrow^* (S, V)$*,* $T\theta \subseteq S$*, and* $V(L\theta) = \top$.

4.1.2 Decidability

Well-moded derivations. In order to obtain a decidability result we will show that when checking the satisfiability of a query, it is correct to restrict the search space by considering only *well-moded* terms. The notion of mode is inspired from [CR06]. It is similar to the idea of having well-typed rules, but we prefer to call them well-moded to emphasize that we do not have a typing assumption. Informally, we do not assume that a device is able discriminate between bitstrings that were generated, *e.g.*, keys and random data. We simply show that if there exists an attack, then there exists an attack where bitstrings are used for the purpose they were originally created for.

Let us illustrate the notion of modes through the following example.

Example 4.2 *We consider the following set of modes:*

$$\mathsf{Mode} = \{\mathsf{Cipher}, \mathsf{Key}, \mathsf{Seed}, \mathsf{Nonce}, \mathsf{Handle}, \mathsf{Attribute}\}.$$

The following rules define the modes of the associated function symbol:

$$\begin{array}{rcll}
\mathsf{h} & : & \mathsf{Nonce} \times \mathsf{Key} \to \mathsf{Handle} & \\
\mathsf{senc} & : & \mathsf{Key} \times \mathsf{Key} \to \mathsf{Cipher} & \\
\mathsf{aenc} & : & \mathsf{Key} \times \mathsf{Key} \to \mathsf{Cipher} & \\
\mathsf{pub} & : & \mathsf{Seed} \to \mathsf{Key} & \\
\mathsf{priv} & : & \mathsf{Seed} \to \mathsf{Key} & \\
\mathsf{att} & : & \mathsf{Nonce} \to \mathsf{Attribute} & \textit{for all } \mathsf{att} \in \mathcal{A} \\
\mathsf{x}_1, \mathsf{x}_2, \mathsf{n}_1, \mathsf{n}_2 & : & \mathsf{Nonce} & \\
\mathsf{y}_1, \mathsf{y}_2, \mathsf{k}_1, \mathsf{k}_2 & : & \mathsf{Key} & \\
\mathsf{z}, \mathsf{s} & : & \mathsf{Seed} &
\end{array}$$

Intuitively, these modes can be interpreted as sorts. The rules described in Figure 4.1 are well-moded w.r.t. the modes described above. This is also the case of the following rules which represent the deduction capabilities of the attacker:

$$\begin{array}{rcl}
\mathsf{y}_1,\ \mathsf{y}_2 & \to & \mathsf{senc}(\mathsf{y}_1, \mathsf{y}_2) \\
\mathsf{senc}(\mathsf{y}_1, \mathsf{y}_2),\ \mathsf{y}_2 & \to & \mathsf{y}_1 \\
\mathsf{y}_1,\ \mathsf{y}_2 & \to & \mathsf{aenc}(\mathsf{y}_1, \mathsf{y}_2) \\
\mathsf{aenc}(\mathsf{y}_1, \mathsf{pub}(\mathsf{z})),\ \mathsf{priv}(\mathsf{z}) & \to & \mathsf{y}_1 \\
\mathsf{aenc}(\mathsf{y}_1, \mathsf{priv}(\mathsf{z})),\ \mathsf{pub}(\mathsf{z}) & \to & \mathsf{y}_1 \\
\mathsf{z} & \to & \mathsf{pub}(\mathsf{z})
\end{array}$$

The term $\mathsf{senc}(\mathsf{senc}(\mathsf{k}_1, \mathsf{k}_2), \mathsf{k}_1)$ *is not well-moded because of its subterm* $\mathsf{senc}(\mathsf{k}_1, \mathsf{k}_2)$.

We say that a derivation is well-moded if each of its states is well-moded. We have shown that it is correct to only consider well-moded derivations.

Theorem 4.1 *Let $\mathcal{R}$ be a set of well-moded rules. Let $q_0 = (S_0, V_0)$ be a well-moded state such that for each mode m, there exists a well-moded term $t_\mathsf{m} \in S_0$ of mode m. Let Q be a well-moded query that is satisfiable. Then there exists a well-moded derivation witnessing this fact.*

We may note that our proof of Theorem 4.1 is constructive: given a derivation we can always transform it into a well-moded one by replacing ill-moded subterms by t_m where m is the expected mode of the replaced subterm.

Note that we do not assume that an implementation enforces well-modedness. We allow an attacker to yield derivations that are not well-moded. Our result however states that whenever there exists an attack that uses a derivation which is not well-moded there exists another attack that is. Our result is arguably even more useful than an implementation that would enforce typing: it seems unreasonable that an implementation could detect whether a block has been encrypted once or multiple times, while our result avoids consideration of multiple encryptions

Unfortunately, Theorem 4.1 by itself is not very informative: it is always possible to have a single mode Msg which implies that all derivations are well-moded. However, the modes used in our modelling of PKCS# 11 imply that all well-moded terms have bounded message length. It is easy to see that well-moded terms have bounded message length whenever the graph on modes is acyclic. Note that for instance a rule which contains nested encryption does not yield a bound on the message size.

However, bounded message length is not sufficient for decidability. Indeed, undecidability proofs [DLM04, TEB05] for security protocols with bounded message length and unbounded number of nonces are easily adapted to our setting. We only need to consider rules of the form $T \xrightarrow{\mathsf{new}\,\tilde{n}} T'$ (no literal) to realize their encodings of the Post Correspondence Problem. Therefore we bound the number of atomic data of each mode, and obtain the following corollary of Theorem 4.1:

Corollary 4.1 *Let $\mathcal{R}$ be a set of well-moded rules such that well-modedness implies a bound on the message length. Let $q_0 = (S_0, V_0)$ be a well-moded state such that for each mode m, there exists a well-moded term $t_{\mathsf{m}} \in S_0$ of mode m. The problem of deciding whether the query Q is satisfiable is decidable when the set of names $\mathcal{N}$ is finite.*

4.1.3 Analysis of PKCS#11

The main application of the previous result is the fragment of PKCS#11 described in Figure 4.1. Thanks to Corollary 4.1, we are able to bound the search space and to realize some experiments with a well-known model-checker, NuSMV, [CCG+02].

PKCS#11 is a standard designed to promote interoperability, not a tightly defined protocol with a particular goal. As such, the aim of our experiments was to analyse a number of different configurations in order to validate our approach. The configuration is defined by a policy which dictates how attributes may be set, in particular indicating conflicting attributes for a key. We take as our security property the secrecy of sensitive keys, stated in the manual as a property of the interface, [RSA04, p. 31]. Roughly speaking, our methodology was to start with a configuration involving only symmetric keys, and to try to restrict the API until a secure configuration was found. We then added asymmetric keypairs, and repeated the process. A secure configuration proved only to be possible by adding the 'trusted keys' mechanism introduced in PKCS#11 v2.20. We have also analyzed proprietary extensions to the standard used by two commercial providers (nCipher and Eracom) of cryptographic hardware.

Our experiments give an idea of how difficult the secure configuration of a PKCS#11 based API is. Although none of the attacks are particularly subtle or complex, there are so many variations that without some formal work, it is very hard to be sure that a certain configuration is secure. We have seen that no matter how many pairs of conflicting attributes are identified, the fact that the attacker can re-import the same key several times means he can always circumvent the restrictions. To prevent this, we investigated three solutions: the trusted keys mechanism, which is part of the standard, and the proprietary measures of nCipher and Eracom.

All experiments are available at `http://www.lsv.ens-cachan.fr/~steel/pkcs11`.

As a follow-up to this work, Steel *et al.* [BCFS10] developed a more mature version of this tool, called Tookan (using SAT-MC as the underlying tool with many optimiza-

tions and automating configuration extraction of tokens). Their tool was able to analyze commercially available PKCS#11 security tokens. In particular Tookan automates a reverse engineering process to extract the configuration of a token in order to analyze it. In [BCFS10] 17 devices were studied: 9 of them were vulnerable to attacks while the 8 other had severely restricted functionality (they did not allow to import/export sensitive keys).

4.2 Analysis of the TPM

4.2.1 An overview of the TPM

The TPM stores cryptographic keys and other sensitive data in its shielded memory, and provides ways for platform software to use those keys to achieve security goals. To store data using a TPM, one creates TPM keys and uses them to encrypt the data. TPM keys are arranged in a tree structure, with the *storage root key* (SRK) at its root. To each TPM key is associated a 160-bit string called *authdata*, which is analogous to a password that authorises use of the key. A user can use a key loaded in the TPM through the interface provided by the device. This interface is actually a collection of commands that (for example) allow one to load a new key inside the device, or to certify a key by another one. All the commands have as an argument an authorisation HMAC that requires the user to provide a proof of knowledge of the relevant authdata. Each command has to be called inside an *authorisation session* which may regroup several commands. The TPM offers two kinds of authorisation sessions: *Object Independent Authorisation Protocol* (OIAP), which creates a session that can manipulate any object, but works only for certain commands, as well as *Object Specific Authorisation Protocol* (OSAP), which creates a session that manipulates a specific object specified when the session is set up. We will not go into the details of these session mechanisms: the main idea is that the sessions use a system of *rolling nonces* to ensure the freshness and the link between these messages. For each message a new nonce is created. Nonces from the user process are called *odd nonces*, and nonces from the TPM are called *even nonces*. Each message includes an hmac on the most recent odd and even nonce.

Commands. The TPM provides a collection of commands that allow one to create new keys and to manipulate them. These commands are described in [TCG07]. To store data using a TPM, one creates a TPM key and uses it to encrypt the data. TPM keys are arranged in a tree structure. The *Storage Root Key* (SRK) is created by a command called TPM_TakeOwnership. At any time afterwards, a user process can create a child key of an existing key. Once a key has been created, it may be loaded, and used for operations requiring a key. We have concentrated on the following four commands:

- TPM_CreateWrapKey generates a new key and wraps (*i.e.*, encrypts) it under a parent key, which is given as an argument. It returns the wrapped key.
- TPM_LoadKey2 takes as argument a wrap (previously created by the command TPM_CreateWrapKey), and returns a handle, that is, a pointer to the key stored in the TPM memory.

- TPM_CertifyKey requires two key handle arguments and returns a key certificate, *i.e.* it signs the public part of one key with the private part of the other key.
- TPM_UnBind allows decryption with the secret key of a stored handle (the encryption is supposed to be performed in software without the help of the TPM).

As an illustration we will detail the TPM_CertifyKey command a bit more. The command requires two key handle arguments, representing the certifying key and the key to be certified, and two corresponding authorisation HMACs. It returns the certificate.

Assume that two OIAP sessions are already established with their current even rolling nonce ne_1 and ne_2 respectively, and that two keys $\mathsf{handle}(auth_1, sk_1)$ and $\mathsf{handle}(auth_2, sk_2)$ are already loaded inside the TPM. The TPM_CertifyKey command can be informally described as follows:

$$\begin{array}{rcl} n, no_1, no_2, & & \mathsf{new}\ Ne_1, \mathsf{new}\ Ne_2 \cdot \\ \mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne_1, no_1 \rangle) & & Ne_1, Ne_2, \mathrm{certif} \\ \mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne_2, no_2 \rangle) & \rightarrow & \mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, Ne_1, no_1, \mathrm{certif} \rangle) \\ \mathsf{handle}(auth_1, sk_1), \mathsf{handle}(auth_2, sk_2) & & \mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, Ne_2, no_2, \mathrm{certif} \rangle) \end{array}$$

where $\mathrm{certif} = \mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$. The intuitive meaning of such a rule is: if an agent (possibly the attacker) has the data items on the left, then, by means of the command, he can obtain the data items on the right. The new keyword indicates that data, *e.g.* nonces or keys, is freshly generated.

The user requests to certify the key $\mathsf{pk}(sk_2)$ with the key sk_1. For this, he generates a nonce n and two odd rolling nonces no_1 and no_2 that he gives to the TPM together with the two authorisation HMACs. The TPM returns the certificate certif. Two authentication HMACs are also constructed by the TPM to accompany the response. The TPM also generates two new even rolling nonces to continue the session.

4.2.2 Modelling the TPM

In this section, we describe some of the modelling choices. Some of them are related to the tool ProVerif that we used to perform our analysis. We chose ProVerif after first experimenting with the AVISPA toolsuite [ABB+05], which provides support for mutable global state. However, of the AVISPA backends that support state, OFMC and CL-AtSe require concrete bounds on the number of command invocations and fresh nonces to be given. It is possible to avoid this restriction using SATMC [FS09], but SATMC performed poorly in our experiments due to the relatively large message length, a known weakness of SATMC. We therefore opted for ProVerif using abstractions to model state, as we explain below. We will explain how the TPM commands as well as its security properties can be formalized in our framework.

Modelling state. One of the difficulties in reasoning about security APIs such as that of the TPM is *non-monotonic state*. If the TPM is in a certain state s, and then a command is successfully executed, then typically the TPM ends up in a state $s' \neq s$. Commands that require it to be in the previous state s will no longer work. Some of the over-approximations made by tools such as ProVerif do not work well with non-monotonic state. For example,

although private channels could be used to represent the state changes, the abstraction of private channels that ProVerif makes prevents it from being able to verify correctness of the resulting specification.

We address these restrictions by introducing the assumption that only one command is executed in each session. This assumption appears to be quite reasonable. Indeed, the TPM imposes the assumption itself whenever a command introduces new authdata. Moreover, tools like TPM/J [Sar] that provide software-level APIs also implement this assumption. Again to avoid non-monotonicity, we do not allow keys to be deleted from the memory of the TPM; instead, we allow an unbounded number of keys to be loaded.

An important aspect of the TPM is its key table that allows one to store cryptographic keys and other sensitive data in its shielded memory. Our aim is to allow the key table to contain dishonest keys, *i.e.* keys for which the attacker knows the authdata, as well as honest keys. Some of these keys may also share the same authdata. Indeed, it would be incorrect to suppose that all keys have distinct authdata, as the authdata may be derived from user chosen passwords. Our first idea was to use a binary function symbol $\mathsf{handle}(auth, sk)$ to model a handle to the secret key sk with authdata $auth$. We use private functions, *i.e.* functions which may not be applied by the attacker, to allow the TPM process to extract the authdata and the secret key from a handle. This models a lookup in the key table where each handle can indeed be associated to its authdata and private key. Unfortunately, with this encoding ProVerif does not succeed in proving some expected properties. The tool outputs a false attack based on the hypothesis that the attacker knows two handles $\mathsf{handle}(auth_1, sk)$ and $\mathsf{handle}(auth_2, sk)$ which are built over two distinct authdata but the same secret key (which is impossible). We therefore use a slightly more involved encoding where the handle depends on the authdata and a *seed*; the secret key is now obtained by applying a binary private function symbol (denoted hsk hereafter) to both the authdata and the seed. Hence, $\mathsf{handle}(auth_1, s)$ and $\mathsf{handle}(auth_2, s)$ will now point to two different private keys, namely $\mathsf{hsk}(auth_1, s)$ and $\mathsf{hsk}(auth_2, s)$. This modelling avoids the above mentioned false attacks.

Modelling commands. In our modelling we have two processes for each command: a user process and a TPM process. The user process (*e.g.* User_CertifyKey) models an honest user who makes a call to the TPM while the TPM process (*e.g.* TPM_CertifyKey) models the TPM itself. The user process first takes parameters, such as the key handles used for the given command, and can be invoked by the adversary. This allows the adversary to schedule honest user actions in an arbitrary way without knowing himself the authdata corresponding to the keys used in these commands.

Our model assumes that the attacker can intercept, inject and modify commands sent by applications to the TPM, and the responses sent by the TPM. While this might not be the case in all situations, it seems to be what the TPM designers had in mind; otherwise, neither the authentication HMACs keyed on existing authdata, nor the encryption of new authdata would be necessary.

Modelling security properties of the TPM The TPM specification does not detail explicitly which security properties are intended to be guaranteed, although it provides some hints. The specification [TCG07, Part I, p.60] states that: "*The design criterion of the*

protocols is to allow for ownership authentication, command and parameter authentication and prevent replay and man in the middle attacks." We will formalise these security properties as *correspondence properties*:

1. If the TPM has executed a certain command, then a user in possession of the relevant authdata has previously requested the command.

2. If a user considers that the TPM has executed a certain command, then either the TPM really has executed the command, or an attacker is in possession of the relevant authdata.

The first property expresses authentication of user commands, and is achieved by the authorisation HMACs that accompany the commands. The second one expresses authentication of the TPM, and is achieved by the HMACs provided by the TPM with its answer. We argue that the TPM certainly aims at achieving these properties, as otherwise there would be no need for the HMAC mechanism.

4.2.3 Analysing the TPM with ProVerif

All the files for our experiments described below are available on line at:

`http://www.lsv.ens-cachan.fr/~delaune/TPM/`

Our methodology was to first study some core key management commands in isolation to analyse the weakness of each command. This leads us to propose some fixes for these commands. Then, we carried out an experiment where we consider the commands TPM_CertifyKey, TPM_CreateWrapKey, TPM_LoadKey2, and TPM_UnBind together. We consider the fixed version of each of these commands and show that the security properties are satisfied for a scenario that allows:

- an attacker to load his own keys inside the TPM, and
- an honest user to use the same authdata for different keys.

In a first serie of experiments, we model the command TPM_CertifyKey in isolation. Then, we model only the command TPM_CreateWrapKey. Lastly, we consider a model where the commands TPM_CertifyKey, TPM_CreateWrapKey, TPM_LoadKey2, and TPM_UnBind are taken into account. In all experiments, the security properties under test are the correspondence properties explained above.

Experiments with TPM_CertifyKey First, we consider a configuration with two keys loaded inside the TPM. The attacker is assumed to know the two handles $\mathsf{handle}(auth_1, sk_1)$ and $\mathsf{handle}(auth_2, sk_2)$. From the handle $\mathsf{handle}(auth_1, sk_1)$, the attacker can obtain the corresponding public key $\mathsf{pk}(sk_1)$. However, he can obtain neither the private key sk_1, nor the authdata $auth_1$ required to manipulate the key through the device. For the moment, we assume that the attacker does not have his own key loaded onto the device.

ProVerif discovers an attack, described (in a slightly simplified form) in Figure 4.2, that comes from the fact that the command involved two keys. The attacker can easily swap

the role of these two keys: he swaps the two HMACs, the two key handles, and the rolling nonces provided in input of the command. Hence, the TPM will output the certificate $\mathsf{cert}(sk_2, \mathsf{pk}(sk_1))$ whereas the user asked for obtaining the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$.

By performing also the swap on the answer provided by the TPM, the attacker can provide two valid HMACs to the user who will accept the wrong certificate. Hence, the second correspondence property is not satisfied. Note that if the user chooses to verify the certificate he received with the checkcert algorithm, then this attack is not valid anymore and ProVerif is able to verify that this second correspondence property holds.

Initial knowledge of Charlie: $\mathsf{handle}(auth_1, sk_1)$, $\mathsf{handle}(auth_2, sk_2)$.

Trace: Charlie swaps the two authorisation HMACs, and swaps the two response HMACs.

USER	$\rightarrow$	TPM	:	request to open two OIAP sessions
TPM	$\rightarrow$	USER	:	ne_1, ne_2
USER	requests key certification to obtain $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$			
USER	$\rightarrow$	Charlie	:	n, no_1, no_2, $\mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne_1, no_1 \rangle)$,$\mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne_2, no_2 \rangle)$
Charlie	$\rightarrow$	TPM	:	n, no_2, no_1, $\mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne_2, no_2 \rangle)$,$\mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne_1, no_1 \rangle)$
TPM	$\rightarrow$	Charlie	:	ne'_1, ne'_2, $\mathsf{cert}(sk_2, \mathsf{pk}(sk_1))$, $\mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne'_1, no_2 \rangle)$, $\mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne'_2, no_1 \rangle)$
Charlie	$\rightarrow$	USER	:	ne'_2, ne'_1, $\mathsf{cert}(sk_2, \mathsf{pk}(sk_1))$, $\mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne'_2, no_1 \rangle)$, $\mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne'_1, no_2 \rangle)$
USER checks the HMACs and accepts the certificate $\mathsf{cert}(sk_2, \mathsf{pk}(sk_1))$.				

Figure 4.2: Attack trace for Experiment 1.

We patch the command TPM_CertifyKey by considering two different tags for the two different HMACs. More precisely, we replace the constant cfk with cfk_1 (resp. cfk_2) in the first (resp. second) HMAC provided by the user and also the one provided by the TPM. The attacks reported in our first experiment are prevented and ProVerif is able to verify the two correspondence properties.

As a further step we add to the initial configuration another key for Alice and assume that this new key sk'_2 has the same authdata as a previous key of Alice already loaded onto the TPM. Hence, in our model, we have that $\mathsf{handle}(auth_1, sk_1)$, $\mathsf{handle}(auth_2, sk_2)$, and $\mathsf{handle}(auth_2, sk'_2)$ are terms known by the attacker Charlie. ProVerif discovers another attack where the attacker can exchange the key handle $\mathsf{handle}(auth_2, sk_2)$ provided by the honest user in entry of the command with another handle having the same authdata, i.e. $\mathsf{handle}(auth_2, sk'_2)$. The TPM will answer by sending the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk'_2))$ instead of $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$.

This attack comes from the fact that the HMAC is only linked to the key via the authdata. Thus, as soon as two keys share the same authdata, this leads to some confusion. A way to fix this would be to add the key handle inside the HMAC, but the TPM designers chose not to do this because they wanted to allow middleware to unload

and reload keys (and therefore possibly change key handles) without the knowledge of application software that produces the HMACs. A more satisfactory solution, that has been proposed for future versions of the TPM, is to add (the digest of) the public key inside the HMAC. Hence, for instance, the HMAC built by the user is now of the form $\mathsf{hmac}(auth, \langle \mathsf{cfk1}, \mathsf{pk}(sk), n, ne_1, no_1 \rangle)$. The same transformation is done on all the HMACs.

The previous attacks do not exist anymore and ProVerif is able to verify that the two correspondence properties hold.

Next, we assume that the attacker has his own key loaded onto the device. This means that he knows a key handle $\mathsf{handle}(auth_i, sk_i)$ and the authdata $auth_i$ that allows him to manipulate sk_i through the device. He has also access to the public key $\mathsf{pk}(sk_i)$. However, he does not know sk_i that is stored inside the TPM. We rediscover the attack of [GRS+07], showing that the attacker can manipulate the messages exchanged between the USER and the TPM in such a way that the TPM will provide the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk_i))$ to a user that has requested the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$.

The attack of [GRS+07] comes from the fact that the attacker can replace the user's HMAC with one of his own (pertaining to his own key). The TPM will not detect this change since the only link between the two HMACs is the nonce n known by the attacker. To fix this, it seems important that each HMAC contains something that depends on the two keys involved in the certificate. So, we add a digest of each public key inside each HMAC. For instance, the first HMAC built by the user will now be of the form:

$$\mathsf{hmac}(auth_1, \langle \mathsf{cfk1}, \mathsf{pk}(sk_1), \mathsf{pk}(sk_2), n, ne_1, no_1 \rangle).$$

The above attack is not possible anymore as the TPM will only accept two HMACs that refer to the same pair of public keys.

ProVerif is now able to verify that the two correspondence properties hold. However, it does not succeed in proving injectivity for the property expressing authentication of the user. This is due to a limitation of the tool and does not correspond to a real attack.

Experiments with TPM_CreateWrapKey We now study the TPM_CreateWrapKey command in isolation. For this command, we need an OSAP session. We consider a configuration with 2 keys pairs $(sk_1, \mathsf{pk}(sk_1))$ and $(sk_2, \mathsf{pk}(sk_2))$ loaded inside the TPM known to the attacker. For the moment, the intruder does not have his own key loaded onto the device.

For this simple configuration, ProVerif is able to verify that the two correspondence properties hold. Note that this command involves only one key, thus the kind of confusion that exists for the TPM_CertifyKey command is not possible on the command TPM_CreateWrapKey.

As a first step we again add another key for Alice to the initial configuration and assume that this new key sk_2' has the same authdata as a previous key of Alice already loaded in the TPM.

We discover an attack of the same type as for TPM_CertifyKey where the attacker replaces $\mathsf{handle}(auth_2, sk_2)$ by $\mathsf{handle}(auth_2, sk_2')$. Hence, at the end, the user will obtain a wrap that is not the expected one. Note that the user cannot detect that the wrap he received has been performed with $\mathsf{pk}(sk_2')$ instead of $\mathsf{pk}(sk_2)$ since he does not have access to the private part of the key.

As in the case of the TPM_CertifyKey command, a way to fix this, is to add pk(sk) inside the HMAC. Then ProVerif is able to verify the two correspondence properties even if we load a 'dishonest' key inside the TPM, *i.e.* a key for which the attacker knows the authdata.

Experiments with the complete model We now consider a much richer scenario. We consider the commands:

- TPM_CertifyKey (the patched version described above to avoid the previous attacks),
- TPM_CreateWrapKey, TPM_LoadKey2, and TPM_UnBind for which we add the public key inside the HMAC (again to avoid the kind of attacks that we described above).

We consider a scenario where an honest key and a dishonest key are already loaded inside the TPM. Using TPM_CreateWrapKey and TPM_LoadKey2, the honest user and the attacker will be able to create and load new keys into the device. Hence, having only two keys loaded in the TPM in the initial configuration is not a restriction. An honest user is allowed to use the same authdata for different keys.

ProVerif is able to establish the 8 correspondence security properties. However, as above, in one case, it is not able to verify the injective version of the property.

4.3 Conclusion and perspectives

In this chapter we described our work on two standardized security APIs, PKCS#11 and the TPM. Both APIs required to model non-monotonic state making these case studies non-trivial for existing tools. In both cases we were able to rediscover known attacks and some non-trivial variants of them. In the case of the TPM identifying the security property which we argue should be guaranteed by the TPM is in itself an interesting contribution.

Most of the existing work on security APIs relied on the experience of the use of formal methods in the analysis of security protocols. While security protocols are similar to APIs, there are some important differences which motivate the development of a dedicated theory for analysing APIs. We give here some concrete topics that we would explore.

Models with storage. As already discussed one difference with most security protocols[1] is that security APIs rely on the storage of some data which is persistent, and can be read or modified by different calls to the API, and which affects the execution of API commands. An example of such a storage are attributes of keys in the PKCS# 11 standard. One of our aims is to develop models and analysis techniques which allow us to consider such a data store which can be read and updated by parallel processes accessing it.

[1] One may note that a similar notion of state also appears in optimistic fair exchange protocols [ASW97] where a trusted third party needs to store the status of a given transaction to ensure responding in a coherent way.

Security properties as ideal systems. As we have noticed in this chapter it is sometimes difficult to define what is the expected security property that a security API should ensure. Therefore, an appealing approach would be to specify the security of APIs by the means of an ideal API. One direction would be to use the framework of universal composability (UC) [Can01]. UC has emerged during the last years as a powerful model for showing the security of complex systems. The main idea is to start with a system which is trivially secure by construction and then refine each of the components towards a realization, generally relying on cryptography. The individual components are generally specified by an interface and it seems natural to use similar ideas in the context of security APIs. We could also build on our recent work on UC in the applied pi calculus [24].

Computational soundness. During recent years there has been a drive to relate symbolic (aka Dolev-Yao) security proofs and computational security proofs [AR02, 12, CC08] (see also Chapter 7). However, the hypotheses which are needed seem to be unrealistic in the context of security APIs. For instance, the hardware module underlying the API may have restricted computational power and needs to execute efficient encryption algorithms that may not satisfy strong security assumptions that are needed for existing soundness results. We wish to investigate how these hypotheses can be weakened in the particular context of security APIs, exploiting the particular structure of the data which is allowed in the requests to the API. This work may also built on recent developments in cryptography, *e.g.* deterministic authenticated encryption [RS06], allowing for instance to securely encrypt keys with their attributes.

CHAPTER 5

Automated verification of equivalence properties

In the Chapter 3 we have described our modelling of privacy type properties in e-voting in terms of equivalence properties. When analysing protocols we were faced with the lack of tools and decision procedures for deciding (or trying to prove) equivalence properties. A notable exception in this area is the tool ProVerif which succeeds in proving some equivalences [BAF05]. However, ProVerif tries to prove a relation which is finer than observational equivalence, called *diff equivalence*. In particular for the examples which arised in e-voting this relation was too fine. Sometimes, we could rely on ProVerif to prove lemmas about static equivalence of particular frames. However, for some equational theories ProVerif was not able to prove static equivalence either and none of the existing decidability results for static equivalence [AC06] did apply.

While we got initially interested in deciding equivalence properties through privacy in electronic voting, equivalence properties are of interest in many other applications. They can for instance be used to model anonymity properties in other areas [DJP10, DDS10, AF04], resistance against offline guessing attacks [CDE05, Bau05, 27], strong flavours of secrecy [Bla04] or indistinguishability between a protocol and an ideal version which can be thought of as the protocol specification [AG99, Can01, BPW07].

In this chapter we describe three results. First, we describe a decision procedure for static equivalence [23, 6] which will be part of Ş. Ciobâcă's PhD thesis work. Then we describe a combination result [25, 8] for static equivalence (for disjoint equational theories), which was part of A. Mercier's PhD thesis. Finally we describe a symbolic semantics [31, 9] for the applied pi calculus which allows to decide a sound approximation of observational equivalence for a fragment of the applied pi calculus.

5.1 A decision procedure for static equivalence

In this section we describe a procedure for the problem of static equivalence which is correct, in the sense that if it terminates it gives the right answer, for any convergent

equational theory. For the remainder of this section we suppose that the equational theory $\mathcal{E}$ can indeed be oriented into a convergent rewrite system $\mathcal{R}_\mathcal{E}$. As deduction and static equivalence are undecidable for this class of equational theories [AC06], the procedure does not always terminate. However, we show that it does terminate for the class of subterm convergent equational theories (already shown decidable in [AC06]) and several other theories among which the theory of trapdoor commitment encountered in our electronic voting case studies. This work is also a part of Ş. Ciobâcă's PhD thesis work and he implemented the procedure in an efficient prototype. Our procedure relies on a simple fixed point computation based on a few saturation rules, making it convenient to implement.

5.1.1 Preliminary definitions

We consider two binary predicates $\rhd$ and $\sim$ on terms, which we write using infix notation. These predicates are interpreted over frames φ as follows:

1. $R \rhd t$ is true whenever R is a recipe for deducing t in φ
2. $U \sim V$ whenever $(U =_\mathcal{E} V)\varphi$

The main data structures of our algorithm are two types of Horn clauses, written here as $[H \Leftarrow \{L_1, \ldots, L_n\}]$ (read as $L_1 \wedge \ldots \wedge L_n$ implies H), which we call *deduction*, respectively *equational facts*.

Definition 5.1 (facts) *A* deduction fact *(resp.* equational fact*) is an expression denoted* $[U \rhd u \Leftarrow \Delta]$ *(resp.* $[U \sim V \Leftarrow \Delta]$*) where* Δ *is a finite set of the form* $\{X_1 \rhd t_1, \ldots, X_n \rhd t_n\}$ *that contains the* hypotheses *of the fact. Moreover, we assume that:*

- $u, t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ *with* $vars(u) \subseteq vars(t_1, \ldots, t_n)$*;*
- $U, V \in \mathcal{T}(\mathcal{F}, \mathcal{N}, \mathcal{X})$ *and* $X_1, \ldots, X_n$ *are distinct variables;*
- $vars(U, V, X_1, \ldots, X_n) \cap vars(u, t_1, \ldots, t_n) = \emptyset$*.*

A fact is solved *if* $t_i \in \mathcal{X}$ $(1 \leq i \leq k)$*. Otherwise, it is* unsolved*. A deduction fact is* well-formed *if it is unsolved or if* $u \notin \mathcal{X}$*.*

For notational convenience we sometimes omit curly braces for the set of hypotheses and write $[U \rhd u \Leftarrow X_1 \rhd t_1, \ldots, X_n \rhd t_n]$. When $n = 0$ we simply write $[U \rhd u]$ or $[U \sim V]$.

We say that two facts are equivalent if they are equal up to bijective renaming of variables. In the following we implicitly suppose that all operations are carried out modulo the equivalence classes. In particular set union will not add equivalent facts and inclusion will test for equivalent facts. Also, we allow *on-the-fly* renaming of variables in facts to avoid variable clashes.

We now introduce the notion of generation of a term t from a set of facts F. A term t is generated with recipe R from a set of facts F if $R \rhd t$ is a consequence of the solved facts in F. Formally, we have:

Definition 5.2 (generation) *Let* F *be a finite set of well-formed deduction facts. A term* t *is* generated by F with recipe R*, written* $\mathsf{F} \vdash^R t$*, if*

1. *either* $t = x \in \mathcal{X}$ *and* $R = x$;
2. *or there exist a solved fact* $[R_0 \rhd t_0 \Leftarrow X_1 \rhd x_1, \ldots, X_n \rhd x_n] \in \mathsf{F}$, *some terms* R_i *for* $1 \leq i \leq n$ *and a substitution* σ *with* $\mathsf{dom}(\sigma) \subseteq vars(t_0)$ *such that* $t = t_0\sigma$, $R = R_0[X_1 \mapsto R_1, \ldots, X_n \mapsto R_n]$, *and* $\mathsf{F} \vdash^{R_i} x_i\sigma$ *for every* $1 \leq i \leq n$.

A term t *is* generated by F, *written* $\mathsf{F} \vdash t$, *if there exists* R *such that* $\mathsf{F} \vdash^R t$.

From this definition follows a simple recursive algorithm for effectively deciding whether $\mathsf{F} \vdash t$, providing also the recipe. Termination is ensured by the fact that $|x_i\sigma| < |t|$ for every $1 \leq i \leq n$. Note that using memoization we can obtain an algorithm in polynomial time.

Example 5.1 *Consider the following set of facts:*

$$\begin{array}{lccll} [& w_1 & \rhd & \mathsf{enc}(b,k) &] & (\mathsf{f}_1) \\ [& b & \rhd & b &] & (\mathsf{f}_2) \\ [& \mathsf{enc}(Y_1,Y_2) & \rhd & \mathsf{enc}(y_1,y_2) & \Leftarrow Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3) \end{array}$$

where w_1 *is a variable in the domain of the frame,* a, b, k *are names, and* Y_1, Y_2, y_1, y_2 *are variables. We have that* $\mathsf{enc}(\mathsf{enc}(b,k),b)$ *is generated with recipe* $\mathsf{enc}(w_1,b)$. *This follows easily by instantiating the hypotheses of* f_3 *with* f_1 *and respectively* f_2.

Given a finite set of equational facts E and terms M, N, we write $\mathsf{E} \models M \sim N$ if $M \sim N$ is a consequence, in the usual first order theory of equality, of

$$\{U\sigma \sim V\sigma \mid [U \sim V \;\Leftarrow\; X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{E}\} \text{ where } \sigma = \{X_i \mapsto x_i\}_{1 \leq i \leq k}.$$

Note that it may be the case that $x_i = x_j$ for $i \neq j$ (whereas $X_i \neq X_j$).

5.1.2 Saturation procedure

We define for each deduction fact f its *canonical form* such that for a solved deduction fact $[R \rhd t \Leftarrow X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ each variable x_i occurs at most once in the hypotheses and at least once in t. Unsolved deduction facts are kept unchanged. This canonical form is useful to characterize the form of solved facts when we prove termination.

A *knowledge base* is a tuple (F, E) where F is a finite set of well-formed deduction facts that are in canonical form and E a finite set of equational facts.

Definition 5.3 (update) *Given a fact* $\mathsf{f} = [R \rhd t \Leftarrow X_1 \rhd t_1, \ldots, X_n \rhd t_n]$ *and a knowledge base* (F, E), the update of (F, E) by f, *written* $(\mathsf{F}, \mathsf{E}) \oplus \mathsf{f}$, *is defined as*

$$\begin{cases} (\mathsf{F} \cup \{\mathsf{f}'\}, \mathsf{E}) & \text{if } \mathsf{f} \text{ is solved and } \mathsf{F} \not\vdash t \quad \text{useful fact} \\ \qquad \text{where } \mathsf{f}' \text{ is the canonical form of } \mathsf{f} & \\ (\mathsf{F}, \mathsf{E} \cup \{[R' \sim R\sigma]\}) & \text{if } \mathsf{f} \text{ is solved and } \mathsf{F} \vdash t \quad \text{redundant fact} \\ \qquad \text{where } \mathsf{F} \vdash^{R'} t \text{ and } \sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\} & \\ (\mathsf{F} \cup \{\mathsf{f}\}, \mathsf{E}) & \text{if } \mathsf{f} \text{ is not solved} \quad \text{unsolved fact} \end{cases}$$

The choice of the recipe R' in the *redundant fact* case is defined by the implementation. While this choice does not influence the correctness of the procedure, it might influence its termination as we will see later. Note that, the result of updating a knowledge base by a (possibly not well-formed and/or not canonical) fact is again a knowledge base. Facts that are not well-formed will be captured by the *redundant fact* case, which adds an equational fact.

The role of the update function is to add facts to the knowledge base, while performing some redundancy elimination. If $\mathsf{F} \not\vdash t$, then the new fact clearly provides interesting information and it is added to the knowledge base. If the new fact is unsolved, it is added anyway (because it might prove useful later on). If the new fact is solved and $\mathsf{F} \vdash t$, then this deduction fact does not provide new information about deducible terms, but it might provide a new recipe for terms we already know deducible. Therefore, an equational fact is added instead, stating that the two recipes are equal provided the required hypotheses are satisfied.

Example 5.2 *Consider the knowledge base formed of the following set* F *of deduction facts*

$$\begin{array}{lcll} [& w_1 & \rhd & \mathsf{enc}(b,k) \quad] & (\mathsf{f}_1)\\ [& b & \rhd & b \quad] & (\mathsf{f}_2)\\ [& \mathsf{enc}(Y_1,Y_2) & \rhd & \mathsf{enc}(y_1,y_2) \quad \Leftarrow Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3) \end{array}$$

and the empty set E *of equational facts.*

We have already seen that $\mathsf{enc}(\mathsf{enc}(b,k),b)$ *is generated by* F *with recipe* $\mathsf{enc}(w_1,b)$*. Updating the knowledge base by* $[w_2 \rhd \mathsf{enc}(\mathsf{enc}(b,k),b)]$ *would result in no modification of the set of deduction facts, since we already know that* $\mathsf{enc}(\mathsf{enc}(b,k),b)$ *is generated. However, a new equational fact* $[w_2 \sim \mathsf{enc}(w_1,b)]$ *would be added to the set of equational facts.*

Initialisation. Given a frame $\varphi = \nu\tilde{n}.\{w_1 \mapsto t_1, \ldots, w_n \mapsto t_n\}$, our procedure starts from an *initial knowledge base* associated to φ and defined as follows:

$$\begin{array}{lll} \mathrm{Init}(\varphi) = & (\emptyset,\emptyset)\\ \bigoplus_{1\le i\le n} & [w_i \rhd t_i]\\ \bigoplus_{n\in fn(\varphi)} & [n \rhd n]\\ \bigoplus_{f\in\mathcal{F}} & [f(X_1,\ldots,X_k) \rhd f(x_1,\ldots,x_k) \Leftarrow X_1 \rhd x_1, \ldots \rhd X_k \rhd x_k] \end{array}$$

Example 5.3 *Consider the equational theory modelling* malleable *encryption*

$$\mathcal{E}_{\mathsf{mal}} = \{\mathsf{dec}(\mathsf{enc}(x,y),y) = x, \mathsf{mal}(\mathsf{enc}(x,y),z) = \mathsf{enc}(z,y)\}.$$

This malleable encryption scheme allows one to arbitrarily change the plaintext of an encryption. This theory certainly does not model a realistic encryption scheme but it yields a simple example of a theory which illustrates well our procedures. In particular all existing decision procedure we are aware of fail on this example. The rewriting system $\mathcal{R}_{\mathcal{E}_{\mathsf{mal}}}$ *obtained by orienting the equations from left to right is convergent.*

Consider the frame $\varphi = \nu a, k.\{w_1 \mapsto \mathsf{enc}(b,k)\}$. *The knowledge base* $\mathrm{Init}(\varphi)$ *is made up of the following deduction facts:*

$$\begin{array}{llllll}
[& w_1 & \rhd & \mathsf{enc}(b,k) &] & (\mathsf{f}_1)\\
[& b & \rhd & b &] & (\mathsf{f}_2)\\
[& \mathsf{enc}(Y_1,Y_2) & \rhd & \mathsf{enc}(y_1,y_2) & \Leftarrow Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_3)\\
[& \mathsf{dec}(Y_1,Y_2) & \rhd & \mathsf{dec}(y_1,y_2) & \Leftarrow Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_4)\\
[& \mathsf{mal}(Y_1,Y_2) & \rhd & \mathsf{mal}(y_1,y_2) & \Leftarrow Y_1 \rhd y_1, Y_2 \rhd y_2] & (\mathsf{f}_5)
\end{array}$$

Saturation. The aim of our saturation procedure is to produce

1. a set of solved deduction facts which have the same set of syntactic consequences as the initial set of deduction facts modulo the equational theory;
2. a set of solved equational facts whose consequences are exactly the equations holding in the frame.

The main part of this procedure consists in saturating the knowledge base $\mathrm{Init}(\varphi)$ by means of the transformation rules described in Figure 5.1. The rule Narrowing is designed to apply a rewriting step on an existing deduction fact. Intuitively, this rule allows us to get rid of the equational theory and nevertheless ensures that the generation of deducible terms is complete. This rule might introduce unsolved hypothesis. The rule F-Solving is then used to instantiate the unsolved hypotheses of an existing deduction fact. Unifying and E-Solving add equational facts which remember when different recipes for the same term exist.

Note that this procedure may not terminate and that the fixed point may not be unique (the $\oplus$ operation that adds a new fact to a knowledge base is not commutative).

We write $\Longrightarrow^*$ for the reflexive and transitive closure of $\Longrightarrow$.

Example 5.4 *Continuing Example 5.3, we illustrate the saturation procedure. We can apply the rule Narrowing on fact* f_4 *and rewrite rule* $\mathsf{dec}(\mathsf{enc}(x,y),y) \to x$, *as well as on fact* f_5 *and rewrite rule* $\mathsf{mal}(\mathsf{enc}(x,y),z) \to \mathsf{enc}(z,y)$ *adding facts*

$$\begin{array}{llllll}
[\mathsf{dec}(Y_1,Y_2) & \rhd & x & \Leftarrow Y_1 \rhd \mathsf{enc}(x,y), Y_2 \rhd y] & (\mathsf{f}_6)\\
[\mathsf{mal}(Y_1,Y_2) & \rhd & \mathsf{enc}(z,y) & \Leftarrow Y_1 \rhd \mathsf{enc}(x,y), Y_2 \rhd z] & (\mathsf{f}_7)
\end{array}$$

The facts f_6 *and* f_7 *are not solved and we can apply the rule F-Solving with* f_1 *adding the facts:*

$$\begin{array}{llllll}
[\mathsf{dec}(w_1,Y_2) & \rhd & b & \Leftarrow Y_2 \rhd k] & (\mathsf{f}_8)\\
[\mathsf{mal}(w_1,Y_2) & \rhd & \mathsf{enc}(z,k) & \Leftarrow Y_2 \rhd z] & (\mathsf{f}_9)
\end{array}$$

Rule Unifying can be used on facts $\mathsf{f}_1/\mathsf{f}_3$, $\mathsf{f}_3/\mathsf{f}_9$ *as well as* $\mathsf{f}_1/\mathsf{f}_9$ *to add equational facts. This third case allows one to obtain* $\mathsf{f}_{10} = [w_1 \sim \mathsf{mal}(w_1,Y_2) \Leftarrow Y_2 \rhd b]$ *which can be solved (using E-Solving with* f_2*) to obtain* $\mathsf{f}_{11} = [w_1 \sim \mathsf{mal}(w_1,b)]$, *etc. When reaching a fixed point,* f_9, f_{11} *and the facts in* $\mathrm{Init}(\varphi)$ *are some of the solved facts contained in the knowledge base.*

We now state the soundness and completeness of our transformation rules.

Narrowing

$\mathsf{f} = [M \rhd C[t] \Leftarrow X_1 \rhd x_1, \ldots, X_k \rhd x_k] \in \mathsf{F},\ l \to r \in \mathcal{R}_{\mathcal{E}}$
with $t \notin \mathcal{X},\ \sigma = \mathrm{mgu}(l,t)$ and $vars(\mathsf{f}) \cap vars(l) = \emptyset$.

$$(\mathsf{F},\mathsf{E}) \Longrightarrow (\mathsf{F},\mathsf{E}) \oplus \mathsf{f}_0$$

where $\mathsf{f}_0 = [M \rhd (C[r])\sigma \Leftarrow X_1 \rhd x_1\sigma, \ldots, X_k \rhd x_k\sigma]$.

F-Solving

$\mathsf{f}_1 = [M \rhd t \Leftarrow X \rhd u, X_1 \rhd t_1, \ldots, X_k \rhd t_k],\ \mathsf{f}_2 = [N \rhd s \Leftarrow Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell] \in \mathsf{F}$
with $u \notin \mathcal{X},\ \sigma = \mathrm{mgu}(s,u)$ and $vars(\mathsf{f}_1) \cap vars(\mathsf{f}_2) = \emptyset$.

$$(\mathsf{F},\mathsf{E}) \Longrightarrow (\mathsf{F},\mathsf{E}) \oplus \mathsf{f}_0$$

where $\mathsf{f}_0 = [M\{X \mapsto N\} \rhd t\sigma \Leftarrow \{X_i \rhd t_i\sigma\}_{1\le i\le k} \cup \{Y_i \rhd y_i\sigma\}_{1\le i\le \ell}]$.

Unifying

$\mathsf{f}_1 = [M \rhd t \Leftarrow X_1 \rhd x_1, \ldots, X_k \rhd x_k],\ \mathsf{f}_2 = [N \rhd s \Leftarrow Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell] \in \mathsf{F}$
with $\sigma = \mathrm{mgu}(s,t)$ and $vars(\mathsf{f}_1) \cap vars(\mathsf{f}_2) = \emptyset$.

$$(\mathsf{F},\mathsf{E}) \Longrightarrow (\mathsf{F},\mathsf{E} \cup \{\mathsf{f}_0\})$$

where $\mathsf{f}_0 = [M \sim N \Leftarrow \{X_i \rhd x_i\sigma\}_{1\le i\le k} \cup \{Y_i \rhd y_i\sigma\}_{1\le i\le \ell}]$.

E-Solving

$\mathsf{f}_1 = [U \sim V \Leftarrow Y \rhd s, X_1 \rhd t_1, \ldots, X_k \rhd t_k] \in \mathsf{E},\ \mathsf{f}_2 = [M \rhd t \Leftarrow Y_1 \rhd y_1, \ldots, Y_\ell \rhd y_\ell\} \in \mathsf{F}$
with $s \notin \mathcal{X}$, $\sigma = \mathrm{mgu}(s,t)$ and $vars(\mathsf{f}_1) \cap vars(\mathsf{f}_2) = \emptyset$.

$$(\mathsf{F},\mathsf{E}) \Longrightarrow (\mathsf{F},\mathsf{E} \cup \{\mathsf{f}_0\})$$

where $\mathsf{f}_0 = [U\{Y \mapsto M\} \sim V\{Y \mapsto M\} \Leftarrow \{X_i \rhd t_i\sigma\}_{1\le i\le k} \cup \{Y_i \rhd y_i\sigma\}_{1\le i\le \ell}]$.

Figure 5.1: Saturation rules

Theorem 5.1 (soundness and completeness) *Let φ be a frame and (F,E) be a saturated knowledge base such that $\mathrm{Init}(\varphi) \Longrightarrow^* (\mathsf{F},\mathsf{E})$. Let $t \in \mathcal{T}(\mathcal{F},\mathcal{N})$ and $\mathsf{F}^+ = \mathsf{F} \cup \{[n \rhd n] \mid n \in fn(t) \setminus bn(\varphi)\}$. We have that:*

1. *For all $M \in \mathcal{T}(\mathcal{F},\mathcal{N} \cup \mathsf{dom}(\varphi))$ such that $fn(M) \cap bn(\varphi) = \emptyset$, we have that*
$$M\varphi =_{\mathcal{E}} t \Leftrightarrow \exists N,\ \mathsf{E} \models M \sim N \text{ and } \mathsf{F}^+ \vdash^N t{\downarrow_{\mathcal{R}_{\mathcal{E}}}}$$
2. *For all $M, N \in \mathcal{T}(\mathcal{F},\mathcal{N} \cup \mathsf{dom}(\varphi))$ such that $fn(M,N) \cap bn(\varphi) = \emptyset$, we have*
$$(M =_{\mathcal{E}} N)\varphi \Leftrightarrow \mathsf{E} \models M \sim N.$$

5.1.3 Procedure for static equivalence

Let φ_1 and φ_2 be two frames. The procedure for checking $\varphi_1 \sim_{\mathcal{E}} \varphi_2$ runs as follows:

1. Apply the transformation rules to obtain (if possible) two saturated knowledge bases $(\mathsf{F}_i,\mathsf{E}_i)$, $i = 1,2$ such that $\mathrm{Init}(\varphi_i) \Longrightarrow^* (\mathsf{F}_i,\mathsf{E}_i)$, $i = 1,2$.
2. For $\{i,j\} = \{1,2\}$, for every solved fact $[M \sim N \Leftarrow X_1 \rhd x_1, \ldots, X_k \rhd x_k]$ in E_i, check if $(M\sigma =_{\mathcal{E}} N\sigma)\varphi_j$ where $\sigma = \{X_1 \mapsto x_1, \ldots, X_k \mapsto x_k\}$.
3. If so return *yes*; otherwise return *no*.

The correctness of this procedure follows nearly directly from Theorem 5.1.

5.1.4 Termination

As already announced the saturation process will not always terminate.

Example 5.5 *Consider the convergent rewriting system consisting of the single rule* $f(g(x)) \to g(h(x))$ *and the frame* $\phi = \nu a.\{w_1 \mapsto g(a)\}$. *We have that*

$$\mathrm{Init}(\varphi) \supseteq \{[w_1 \rhd g(a)],\ [f(X) \rhd f(x) \Leftarrow X \rhd x]\}.$$

By Narrowing *we can add the fact* $\mathsf{f}_1 = [f(X) \rhd g(h(x)) \Leftarrow X \rhd g(x)]$. *Then we can apply* F-Solving *to solve its hypothesis* $X \rhd g(x)$ *with the fact* $[w_1 \rhd g(a)]$ *yielding the solved fact* $[f(w_1) \rhd g(h(a))]$. *Applying iteratively* F-Solving *on* f_1 *and the newly generated fact, we generate an infinity of solved facts of the form* $[f(\dots f(w_1)\dots) \rhd g(h(\dots h(a)\dots))]$. *Intuitively, this happens because our symbolic representation is unable to express that the function* h *can be nested an unbounded number of times when it occurs under an application of* g.

We have shown that our procedure terminates on the class of subterm convergent equational theories [AC06], and different particular theories: malleable encryption (defined in Example 5.4), a theory for trap-door commitment ($\mathcal{E}_{\mathsf{td}}$) encountered when studying electronic voting protocols [13], a theory of blind signatures ($\mathcal{E}_{\mathsf{blind}}$) which we introduced first in [37] and which has also been studied in [AC06], as well as theories for addition ($\mathcal{E}_{\mathsf{add}}$) and homomorphic encryption ($\mathcal{E}_{\mathsf{hom}}$) which have been studied in [AC06]. We may note that for the procedure to terminate on the theory modelling homomorphic encryption we need to rely on a particular saturation strategy, which we require to be *fair*.

5.1.5 The KiSs tool

A C++ implementation of the procedures described here is provided in the KiSs (Knowledge in Security protocols) tool [Cio09].

The tool implements a partially fair saturation strategy which is sufficient to decide the theory $\mathcal{E}_{\mathsf{hom}}$. Moreover the tool implements several optimizations described in [6, Section 6.1]. This makes the procedure terminate in polynomial time for subterm convergent equational theories, and the theories $\mathcal{E}_{\mathsf{blind}}$, $\mathcal{E}_{\mathsf{mal}}$ and $\mathcal{E}_{\mathsf{td}}$.

5.1.6 Related work

Several papers study the decision of static equivalence. Most of these results introduce a new procedure for each particular theory and even in the case of the general decidability criterion given in [AC06, CD07], the algorithm underlying the proof has to be adapted for each particular theory, depending on how the criterion is fulfilled. A combination result was obtained in [ACD07]: if deduction and static equivalence are decidable for two disjoint equational theories, then deduction and static equivalence are also decidable for the union of the two theories.

The first generic algorithm that has been proposed handles subterm convergent equational theories [AC06] and covers the classical theories for encryption and signatures. This result is encompassed by the recent work of Baudet *et al.* [BCD09] in which the authors

propose a generic procedure that works for any convergent equational theory, but which may fail or not terminate. This procedure has been implemented in the YAPA tool [Bau08] and has been shown to terminate without failure in several cases (e.g. subterm convergent theories and blind signatures). However, due to its simple representation of deducible terms (represented by a finite set of *ground* terms), the procedure fails on several interesting equational theories like the theory of trapdoor commitments. Our representation of deducible terms overcomes this limitation by including terms with variables which can be substituted by any deducible terms. The performances of the KiSs tool are comparable to the YAPA tool [Bau08, BCD09] and on most examples the tool terminates in less than a second. In [BCD09] a family of contrived examples is presented to diminish the performance of YAPA, exploiting the fact that YAPA does not implement DAG representations of terms and recipes, as opposed to KiSs. As expected, KiSs indeed performs better on these examples. In [BCD09] a class of equational theories for which YAPA terminates is identified and it is not known whether our procedure terminates on this specific class. However, we have shown that our procedure terminates on all examples of equational theories presented in [BCD09]. This requires to prove termination of our saturation procedure for each equational theory presented in [BCD09]. In addition, our tool terminates on the theories $\mathcal{E}_{\mathsf{mal}}$ and $\mathcal{E}_{\mathsf{td}}$ whereas YAPA does not. Of course, YAPA may also terminate on examples outside the class exhibited in [BCD09]. Hence the question whether termination of our procedures encompasses termination of YAPA is still open. Independently of our work, specific decision procedures for the theory of trapdoor commitment and that of reencryption have been presented in [BBRC09].

Another tool that can be used to check static equivalence is ProVerif [Bla01, BAF05]. This tool can handle various equational theories and analyse security protocols under active adversaries. However, termination is not guaranteed in general and the tool performs some safe approximations.

5.2 Reducing equational theories for the decision of static equivalence

Our goal was to develop combination results for the decision of static equivalence for non-disjoint equational theories. As discussed above, a combination result for disjoint equational theories was presented in [ACD07]. We here exhibit criteria on equational theories allowing simplifications for the decision of static equivalence. The kind of simplification we describe is the removal of a particular symbol which we call a valve. This allows us to reduce a static equivalence problem modulo some equational theory to some other static equivalence problems modulo simpler equational theories. We will illustrate our result on a running example, a theory for bilinear pairings for which we obtain a deducibility result.

This work was part of A. Mercier's PhD thesis and was published in [25, 8].

5.2.1 Running example

We will illustrate our specific definitions and lemmas by a running example involving two distinct algebraic groups G_1 and G_2 and a pairing operation e mapping two elements of G_1 to an element of G_2. A concrete definition in a cryptographic setting can be found

in [BF01]. In general, a pairing operation maps elements of an additive group to elements of a multiplicative group in the following way.

$$\begin{array}{rrcl} e: & G_1 \times G_1 & \to & G_2 \\ & e(ag_1, bg_2) & = & e(g_1, g_2)^{ab} \end{array}$$

In some protocols, e.g. [Jou00], one has in fact $g_1 = g_2$. We use this assumption in order to simplify our notations. Moreover, we use a multiplicative notation to represent elements of G_1, e.g. we write $exp_1(x)$ for both xg_1 and xg_2.

Let $\mathcal{S}_{\mathsf{bp}}$ be the set of sorts $\{R, G_1, G_2\}$, R is the sort of the exponents of a chosen generator of the G_i, and G_1 (resp. G_2) are the sorts of the elements of the groups G_1 (resp. G_2). Let $\mathcal{F}_{\mathsf{bp}}$ be the following set of symbols:

$+, \cdot :$	$R \times R \to R$		addition, multiplication of exponents
$-:$	$R \to R$		inverse of exponents
$0_R, 1_R :$	R		constant exponents
$exp_i :$	$R \to G_i$	$i \in \{1,2\}$	exponentiation
$*_i :$	$G_i \times G_i \to G_i$	$i \in \{1,2\}$	multiplication in G_i
$e :$	$G_1 \times G_1 \to G_2$		pairing

The properties of these function symbols are defined by the following equational theory $\mathcal{E}_{\mathsf{bp}}$.

$$\begin{array}{rcl@{\qquad}rcl} x + y & = & y + x & 0_R + x & = & x \\ (x + y) + z & = & x + (y + z) & x + (-x) & = & 0_R \\ x \cdot y & = & y \cdot x & x \cdot (y + z) & = & (x \cdot y) + (x \cdot z) \\ (x \cdot y) \cdot z & = & x \cdot (y \cdot z) & 1_R \cdot x & = & x \end{array}$$

$$exp_i(x) *_i exp_i(y) = exp_i(x + y) \qquad i \in \{1,2\}$$

$$e(exp_1(x), exp_1(y)) = exp_2(x \cdot y)$$

This signature and this equational theory represent operations realized in protocols where the exchanged messages are elements of the groups G_i. The symbol e represents a pairing operation.

Example 5.6 *Bilinear pairing is a central primitive of the Joux protocol [Jou00], a three participant variation of the Diffie-Hellman protocol. It implicitly relies on the decisional Bilinear Diffie-Hellman Assumption (BDH) which can be formally modelled using static equivalence as follows:*

$$\nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c), y_1 \mapsto exp_2(a \cdot b \cdot c)\}$$
$$\sim_{\mathcal{E}_{\mathsf{bp}}}$$
$$\nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c), y_1 \mapsto exp_2(r)\}$$

5.2.2 Valves and reducibility

Our main result concerns signatures involving a special function symbol which we call a *valve*. Intuitively, as it is suggested by the name, a valve f is a symbol that allows to go into one direction, but such that there is no way to go back. More exactly, a valve f takes arguments of sorts $s_1, \ldots, s_k$, and yields a result of sort s, such that no term of sort s_i has a subterm of sort s.

We borrow here some useful notions from graph theory.

Definition 5.4 (Signature graph) *Let $(\mathcal{S}, \mathcal{F})$ be a sorted signature. The graph $\mathcal{G}(\mathcal{S}, \mathcal{F})$ is the directed labelled graph (V, E) where $V = \mathcal{S}$ and $E \subseteq V \times V \times \mathcal{F}$. We write $r \xrightarrow{f} s$ for $(r, s, f) \in E$ and require that $arity(f) = s_1 \times \ldots \times s_k \to s$ and $s_i = r$ for some i.*

We recall that a *path* in a graph is a sequence of edges such that for two consecutive edges $r \xrightarrow{f} s$ and $r' \xrightarrow{f'} s'$ we have $s = r'$. When S_1 and S_2 are sets of vertices we say that there exists a path from S_1 to S_2 iff there exist $s_1 \in S_1$, $s_2 \in S_2$ such that there is a path from s_1 to s_2.

Definition 5.5 (valve) *A symbol f of arity $s_1 \times \ldots \times s_k \to s$ is a* valve *iff*

1. *for every path π from $\{s_1, \ldots, s_k\}$ to $\{s\}$ in $\mathcal{G}(\mathcal{S}, \mathcal{F})$ there is a j, $1 \leq j \leq k$, such that π contains $s_j \xrightarrow{f} s$;*
2. *and there is no path from $\{s\}$ to $\{s_1, \ldots, s_k\}$.*

In other words, f is a valve iff every path from $\{s_1, \ldots, s_k\}$ to $\{s\}$ contains exactly one f. When f of arity $s_1 \times \ldots \times s_k \to s$ is a valve, we also say that f is a valve from $\{s_1, \ldots, s_k\}$ to s.

Example 5.7 *Let us consider the sorted signature $(\mathcal{S}_{\mathsf{bp}}, \mathcal{F}_{\mathsf{bp}})$ introduced in our running example in Section 5.2.1. The signature graph $\mathcal{G}(\mathcal{S}_{\mathsf{bp}}, \mathcal{F}_{\mathsf{bp}})$ is given in Figure 5.2. The symbol e is a valve from $\{G_1\}$ to $\{G_2\}$ as $G_1 \xrightarrow{e} G_2$ lies on every path from $\{G_1\}$ to $\{G_2\}$, and since no path leads from $\{G_2\}$ to $\{G_1\}$. We also have that exp_1 is a valve from $\{R\}$ to $\{G_1\}$. However, exp_2 is not a valve from $\{R\}$ to $\{G_2\}$ as the sequence $R \xrightarrow{exp_1} G_1, G_1 \xrightarrow{e} G_2$ is a path from $\{R\}$ to $\{G_2\}$ not involving exp_2.*

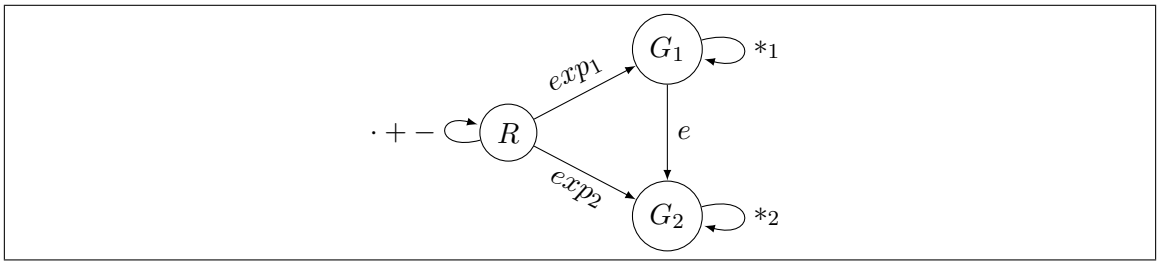


Figure 5.2: The signature graph $\mathcal{G}(\mathcal{S}_{\mathsf{bp}}, \mathcal{F}_{\mathsf{bp}})$.

We are now able to present the central notion of reducibility.

Definition 5.6 (reducible) *Let f be a valve of arity $s_1 \times \ldots \times s_k \to s$. An equational theory $\mathcal{E}$ is* reducible *for f iff for every $n \geq 0$ and sorts $r_1, \ldots, r_n \in \{s_1, \ldots, s_k\}$ there exist m public contexts $T_1[x_1, \ldots, x_n], \ldots, T_m[x_1, \ldots, x_n]$ of arity $r_1 \times \ldots \times r_n \to s$ such that for all k public contexts $C_i[x_1, \ldots, x_n]$ of arity $r_1 \times \cdots \times r_n \to s_i$ with $1 \leq i \leq k$ there exists a public context $D[y_1, \ldots, y_m]$ of arity $s \times \cdots \times s \to s$ such that for any ground terms $t_1, \ldots, t_n$ of sort $r_1, \ldots, r_n$ respectively we have that*

$$f(C_1, \ldots, C_k)[t_1, \ldots, t_n] =_E D[T_1, \ldots, T_m][t_1, \ldots, t_n]$$

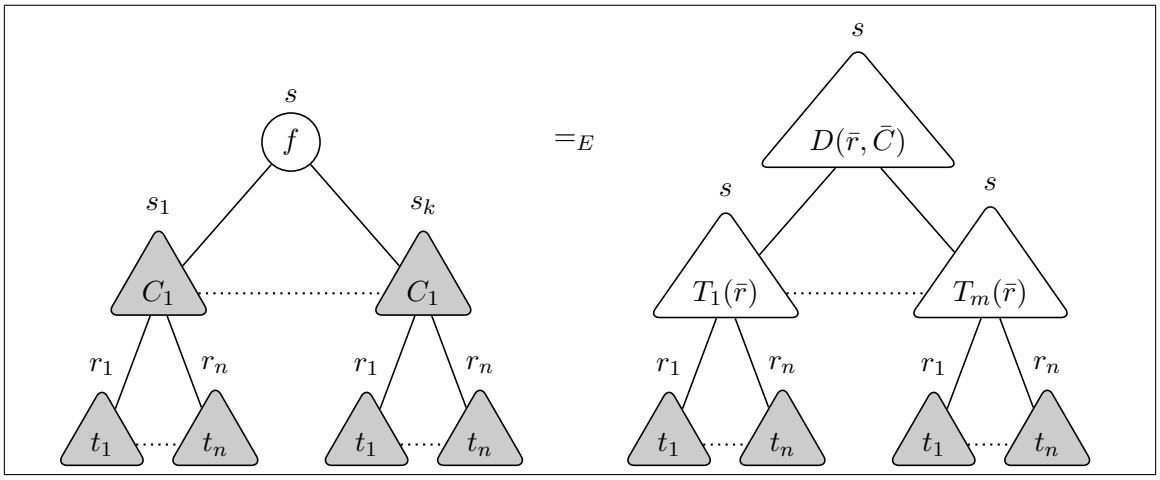


Figure 5.3: Reducibility. Grey objects are of sort $s_1, \ldots, s_n$, while clear objects are of sort s. The choice of the T_i only depends on the sorts $\bar{r} = r_1, \ldots, r_n$, while the choice of D depends both on the contexts $\bar{C} = C_1, \ldots, C_k$ and the sorts $\bar{r}$.

Intuitively, reducibility for a valve f means that given a vector $(r_1, \ldots, r_n)$ of sorts that are all argument sorts of f, there are finitely many maps T_i from $r_1 \times \ldots \times r_n$ to s such that any computation of the form $f(C_1, \ldots, C_k)$ can be simulated by some D entirely inside the sort s by making use of the maps T_i. The crucial point is that the contexts T_i depend only on the sorts $r_1, \ldots, r_n$ but *not* on the contexts C_i. A pictorial view of this definition is given in Figure 5.3.

Proposition 5.1 *$\mathcal{E}_{\mathsf{bp}}$ is reducible for e if the set of names of sort G_1 $\mathcal{N}_{G_1} = \emptyset$.*

Indeed, for any integer n we can define $m = n + \frac{n*(n+1)}{2}$ contexts

$$\begin{array}{lcll} T_i & = & e(x_i, exp_1(1_R)) & \text{for } 1 \leq i \leq n \\ T_{ij} & = & e(x_i, x_j) & \text{for } 1 \leq i \leq j \leq n \end{array}$$

which satisfy the conditions of Definition 5.6.

We now define the reduction of a frame.

Definition 5.7 (reduction) *Let the equational theory $\mathcal{E}$ be* reducible *for f, where f is a valve from A to s, and let $\phi = \nu\tilde{n}\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ be an A-sorted frame. The* reduction *of ϕ is defined as $\overline{\phi} = \nu\tilde{n}\{y_1 \mapsto T_1[t_1, \ldots t_n], \ldots, y_m \mapsto T_m[t_1, \ldots t_n]\}$ where T_i are contexts as defined in Definition 5.6.*

We note that $\overline{\phi}$ is $\{s\}$-sorted.

Example 5.8 *Let ϕ_{BDH} be the G_1-restriction of the frames presented in Example 5.6: $\phi_{BDH} = \nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c)\}$. Using the set of terms T_i and T_{ij} defined above, we get*

$$\begin{array}{lll} \overline{\phi}_{BDH} = \nu a, b, c, r.\{ & y_1 \mapsto e(exp_1(a), exp_1(1)), & y_2 \mapsto e(exp_1(b), exp_1(1)), \\ & y_3 \mapsto e(exp_1(c), exp_1(1)), & y_{11} \mapsto e(exp_1(a), exp_1(a)), \\ & y_{12} \mapsto e(exp_1(a), exp_1(b)), & y_{13} \mapsto e(exp_1(a), exp_1(c)), \\ & y_{22} \mapsto e(exp_1(b), exp_1(b)), & y_{23} \mapsto e(exp_1(b), exp_1(c)), \\ & y_{33} \mapsto e(exp_1(c), exp_1(c)) & \} \end{array}$$

5.2.3 Getting rid of reducible symbols

We now show that if an equational theory $\mathcal{E}$ is *reducible* for f then it is possible to get rid of f when deciding static equivalence. In the we only consider signatures $(\mathcal{S}, \mathcal{F})$ where $f \in \mathcal{F}$ is a valve from $\{s_1, \ldots, s_n\}$ to s such that in $\mathcal{G}(\mathcal{S}, \mathcal{F})$ the only sorts accessible from $\{s_1, \ldots, s_n\}$ are $\{s_1, \ldots, s_n, s\}$. Consequently, the only sort accessible from s is s. This is sufficient to cover our running example and simplifies the presentation of our results.

In order to eliminate f from the equational theory we require additional properties of the equational theory.

Definition 5.8 (sufficiently complete [Com94]) *$\mathcal{E}$ is a* sufficiently complete *equational theory with respect to f if for every ground term $t \in \mathcal{T}(\mathcal{F} \uplus \{f\}, \mathcal{N})$ there exists a ground term $u \in \mathcal{T}(\mathcal{F}, \mathcal{N})$ such that $t =_{\mathcal{E}} u$.*

Definition 5.9 (sufficient equational theory) *Let $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ be a sorted signature and $\mathcal{E}$ an equational theory. An equational theory $\mathcal{E}'$ is sufficient for $\mathcal{E}$ without f iff for any terms $u, v \in T(\mathcal{F}, \mathcal{N})$, $u =_{\mathcal{E}} v$ iff $u =_{\mathcal{E}'} v$ and $\mathcal{E}'$ does not involve f.*

We denote by $\mathcal{E}^{-A}$ the equational theory $\mathcal{E}$ without equations of sort in A.

Theorem 5.2 *Let $\mathcal{E}$ be an equational theory on the sorted signature $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ such that*

- *f is a valve,*
- *$\mathcal{E}$ is a reducible equational theory for f, and*
- *$\mathcal{E}$ is sufficiently complete w.r.t. f.*

If there exists an equational theory $\mathcal{E}'$ sufficient for $\mathcal{E}$ without f then for any $\{A, s\}$-sorted frames ϕ_1 and ϕ_2, we have that

$$\phi_1 \sim_{\mathcal{E}} \phi_2 \quad \text{iff} \quad \phi_{1|A} \sim_{\mathcal{E}'^{-s}} \phi_{2|A} \quad \text{and} \quad \overline{\phi_{1|A}}\phi_{1|s} \sim_{\mathcal{E}'^{-A}} \overline{\phi_{2|A}}\phi_{2|s}$$

5.2.4 A criterion for sufficient equational theories

In this section we make a first attempt to find sufficient criteria for applying Theorem 5.2. Future work includes finding broader criteria. We also briefly explain how our running example fits this criterion.

Definition 5.10 (decomposition) *A pair $(\mathcal{R}, \mathcal{E}')$ is a* decomposition *of an equational theory $\mathcal{E}$ iff*

- *$\mathcal{E}'$ is an equational theory,*
- *$\mathcal{R}$ is a rewriting system convergent modulo $\mathcal{E}'$,*
- *for any terms u and v $u =_{\mathcal{E}} v$ iff $u \downarrow_{\mathcal{R}/\mathcal{E}'} = v \downarrow_{\mathcal{R}/\mathcal{E}'}$.*

Definition 5.11 (exclusively define) *Let $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ be a sorted signature. A rewriting system $\mathcal{R}$* exclusively defines *f if any term in normal form modulo $\mathcal{R}/\mathcal{E}'$ is in $\mathcal{T}(\mathcal{F}, \mathcal{N})$ and if for any rewrite rule $l \to r \in \mathcal{R}$, f appears in l.*

Lemma 5.1 *Let $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ be a signature. If a theory $\mathcal{E}$ on this signature has a decomposition $(\mathcal{R}, \mathcal{E}')$ and if $\mathcal{R}$ exclusively defines f then $\mathcal{E}'$ is sufficient for $\mathcal{E}$ without f.*

Example 5.9 (continued) *We define $\mathcal{R}_{\mathsf{bp}}$ to be the rewriting system obtained by orienting the rule $e(exp_1(x), exp_1(y)) = exp_2(x \cdot y)$ from left to right, and $\mathcal{E}'_{\mathsf{bp}}$ the equational theory $\mathcal{E}_{\mathsf{bp}}$ without this rule. We remark that $(\mathcal{R}, \mathcal{E}'_{\mathsf{bp}})$ is a decomposition of $\mathcal{E}_{\mathsf{bp}}$ and it is easy to see that $\mathcal{R}_{\mathsf{bp}}$ exclusively defines e.*

Corollary 5.1 *If the sets of names of sorts G_1 and G_2 are empty then static equivalence for $\mathcal{E}_{\mathsf{bp}}$ is decidable for $\{G_1, G_2\}$-sorted frames.*

Removing the function symbol e we reduce the decision of static equivalence for $\mathcal{E}_{\mathsf{bp}}$ to deciding static equivalence for two theories that correspond both to the classical equational theory modelling Diffie-Hellman. This problem is is known to be decidable [32] for frames whose only names are of sort R.

5.3 Symbolic bisimulation for the applied pi calculus

In this section we describe our results on observational equivalence, which models indistinguishability in the presence of an active adversary. One of the difficulties in automating proofs of observational equivalence is the infinite number of possible behaviours of the attacker, even in the case that the protocol process itself is finite, *i.e.*, without replication. When the process requests an input from the environment, the attacker can give any term which can be constructed from the terms it has learned so far in the protocol, and therefore the execution tree of the process is potentially infinite-branching. To address this problem, researchers have proposed *symbolic abstractions* of processes, in which terms input from the environment are represented as symbolic variables, together with some constraints. These constraints describe the knowledge of the attacker (and therefore, the range of possible values of the symbolic variables) at the time the input was performed. Reachability properties and also existence of off-line guessing attacks can be verified by solving the constraint systems arising from symbolic executions (e.g. [ALV02b, Bau07]). Our idea was to design symbolic semantics for the applied pi calculus and rely on Baudet's decision procedure for deciding equivalence of constraint systems, *i.e.*, whether two constraint systems have the same set of solutions (as opposed to the classical satisfiability where one checks the existence of a solution). Symbolic methods have already been used in the case of observational equivalence or bisimulation properties in classical process algebras (e.g. [HL95, BN96]). In particular, Borgström *et al.* [BBN04] have defined a sound symbolic bisimulation for the spi calculus. Defining symbolic semantics for the applied pi calculus has shown to be surprisingly difficult technically. In this section we will not describe our symbolic semantics in detail but rather discuss the difficulties that we encountered and the approach we took to circumvent them. A complete description of the symbolic semantics can be found in [31, 9].

5.3.1 The problem of designing a sound and complete symbolic structural equivalence

A natural first step seems to define a symbolic structural equivalence ($\equiv_s$) which is sound and complete in the following (informal) sense:

> *Soundness*: $P_s \equiv_s Q_s$ implies for any valid instantiation σ, $P_s\sigma \equiv Q_s\sigma$;
> *Completeness*: $P_s\sigma \equiv Q$ implies $\exists\, Q_s$ such that $P_s \equiv_s Q_s$ and $Q_s\sigma = Q$.

However, it seems difficult to achieve this. Consider the process

$$P = \mathsf{in}(c,x).\mathsf{in}(c,y).\mathsf{out}(c,f(x)).\mathsf{out}(c,g(y))$$

which can be reduced to

$$P' = \mathsf{out}(c,f(M_1)).\mathsf{out}(c,g(M_2))$$

where M_1 and M_2 are two arbitrary terms provided by the environment. When $f(M_1) =_{\mathcal{E}} g(M_2)$, *i.e.* $f(M_1)$ and $g(M_2)$ are equal modulo the equational theory, we have that $P' \equiv \nu z.(\mathsf{out}(c,z).\mathsf{out}(c,z) \mid \{^{f(M_1)}/_z\})$, but this structural equivalence does not hold whenever $f(M_1) \neq_{\mathcal{E}} g(M_2)$.

The symbolic process $P'_s = \mathsf{out}(c,f(x)).\mathsf{out}(c,g(y))$ has to represent the different cases where $f(x)$ and $g(y)$ are equal or not. Hence, the question of whether the structural equivalence $P'_s \equiv_s \nu z.(\mathsf{out}(c,z).\mathsf{out}(c,z) \mid \{^{f(x)}/_z\})$ is valid cannot be decided, as it depends on the concrete values of x and y. Therefore, symbolic structural equivalence cannot be both sound and complete. This seems to be an inherent problem and it propagates to internal and labelled reduction, since they are closed under structural equivalence.

The absence of sound and complete symbolic structural equivalence, mentioned above, significantly complicates the proof of our main result given in [31, 9]. We therefore split this proof into two parts:

1. We define a more restricted *intermediate* semantics which will provide an intermediate representation of applied pi calculus processes and define an intermediate labelled bisimulation which coincides with the original one.

2. We define a symbolic semantics which is sound and complete w.r.t. the intermediate semantics. Based on the symbolic semantics we define a sound symbolic bisimulation.

5.3.2 Intermediate semantics

Intermediate processes are a selected (but sufficient) subset of the original processes. One may think of them as being processes in some kind of normal form. They only have name restriction (no variable restriction) and all restrictions have to be in front of the process. They have to be *name and variable distinct* (nv-distinct) meaning that we have to use different names (resp. variables) to represent free and bound names (resp. variables), and also that any name (resp. variable) is at most bound once. Moreover, we require an intermediate process to be *applied* meaning that each variable in the domain of the process occurs only once. For instance, the process $A{\downarrow} = \nu b.\mathsf{in}(c,y).\mathsf{out}(a,f(b))$ is the intermediate process associated to the process $A = \nu x.(\mathsf{in}(c,y).\nu b.\mathsf{out}(a,x) \mid \{^{f(b)}/_x\})$.

In the symbolic semantics we will keep the constraint systems separate from the process. This allows us to have a clean division between the bisimulation and the constraint solving part. A side-effect of the separation between the processes and the constraint system is that we forbid α-conversion on symbolic processes as we lose the scope of names in the constraint system, but allow explicit renaming when necessary. We will already forbid α-conversion in the intermediate semantics using naming environments. A naming environment N is a function which maps each variable and name to one of n, f, b (standing for "new" (*i.e.*, fresh), "free" and "bound" respectively). An intermediate process is a pair $(A \; ; \mathsf{N})$ where A is an intermediate extended process and N a naming environment, compatible with A. By compatible we mean that bound names in A should indeed be declared to be bound by N, *etc.*

To show soundness and completeness of the relations $\equiv_i, \rightarrow_i, \xrightarrow{\alpha}_i$ defining the intermediate semantics we introduce the relation $\approx$ on intermediate processes. Intuitively, $\approx$ captures the structural equivalences that are "missing" in $\equiv_i$ with respect to $\equiv$.

Definition 5.12 ($\approx$) *We define $\approx$ to be the smallest equivalence relation on intermediate processes closed under bijective renaming of bound names and variables and such that*

$$\begin{array}{lrcll} \textsc{New-N}_i & (\nu\tilde{n}.\nu m.A \; ; \mathsf{N}) & \approx & (\nu\tilde{n}.A \; ; \mathsf{N}) & \text{if } m \notin fn(A) \\ \textsc{Rew-N}_i & (A\{^M/_x\} \; ; \mathsf{N}) & \approx & (A\{^N/_x\} \; ; \mathsf{N}) & \text{if } M =_{\mathcal{E}} N \end{array}$$

This enables us to show soundness and completeness of $\equiv_i, \rightarrow_i, \xrightarrow{\alpha}_i$ up to $\approx$.

Based on the intermediate semantics we can define an intermediate labelled bisimulation $\approx_i$. Using the soundness and completeness of the semantics we show that $\approx_i$ coincides with the original labelled bisimulation.

Theorem 5.3 *Let A and B be two nv-distinct extended processes and N be a naming environment compatible with $A{\downarrow}$ and $B{\downarrow}$. We have that*

$$A \approx B \text{ if and only if } (A{\downarrow} \; ; \mathsf{N}) \approx_i (B{\downarrow} \; ; \mathsf{N})$$

5.3.3 Constraint systems

Before introducing our symbolic semantics we first define constraint systems.

Definition 5.13 (constraint system) *A* constraint system $\mathcal{C}$ *is a set of constraints where every constraint is of one of the following forms:*

- *"$\varphi \Vdash x$", where $\varphi = \nu\tilde{u}.\sigma$ for some tuple of names and variables $\tilde{u}$ and some substitution σ, and x is a variable which does not appear under a restriction of any frame nor in the domain of any frame;*
- *"$M = N$", where M and N are terms;*
- *"$M \neq N$", where M and N are terms;*
- *"$\mathsf{gd}(M)$" where M is a term;*

The constraint $\varphi \Vdash x$ is useful for specifying the information φ held by the environment when it supplies an input x, *i.e.*, the environment may only instantiate x by a term that is deducible from φ. The constraint $\mathsf{gd}(M)$ means that the term M is ground. We denote by $\mathcal{D}ed(\mathcal{C})$ the *deducibility constraints* of $\mathcal{C}$, i.e. $\{\varphi \Vdash x \mid \text{“}\varphi \Vdash x\text{”} \in \mathcal{C}\}$. When $\mathcal{D}ed(\mathcal{C}) = \{\varphi_1 \Vdash x_1, \ldots, \varphi_\ell \Vdash x_\ell\}$, we define $cv(\mathcal{C}) = \{x_1, \ldots, x_\ell\}$ to be the *constraint variables* of $\mathcal{C}$.

The constraint systems that we consider arise while executing symbolic processes. We therefore restrict ourselves to *well-formed* constraint systems, capturing the fact that the knowledge of the environment always increases along the execution: we allow it to use more names and variables (less restrictions) or give it access to more terms (larger substitution). The fact that the constraint system is not arbitrary is useful when solving the constraints such as in [MS01].

We say that two well-formed constraint systems $\mathcal{C}$ and $\mathcal{C}'$ have *same basis* if $\mathcal{D}ed(\mathcal{C}) = \{\varphi_1 \Vdash x_1, \ldots, \varphi_\ell \Vdash x_\ell\}$ and $\mathcal{D}ed(\mathcal{C}') = \{\varphi'_1 \Vdash x'_1, \ldots, \varphi'_\ell \Vdash x'_\ell\}$ are such that $x_i = x'_i$ and $\mathsf{dom}(\varphi_i) = \mathsf{dom}(\varphi'_i)$ for $1 \leq i \leq \ell$.

Given a well-formed constraint-system $\mathcal{C}$ such that $\mathcal{D}ed(\mathcal{C}) = \{\varphi_1 \Vdash x_1, \ldots, \varphi_\ell \Vdash x_\ell\}$ an *$\mathcal{E}$-solution* of $\mathcal{C}$ is a substitution θ whose domain is $cv(\mathcal{C})$ which satisfies the constraints of $\mathcal{C}$. We require that $vars(x_i\theta) \cap (\mathsf{dom}(\varphi_\ell) \setminus \mathsf{dom}(\varphi_i)) = \emptyset$, to ensure that the environment does not use information that will only be revealed “in the future”; it can use only the entries of the frame that have previously been added. Moreover, we require $names(x_i\theta) \cap \tilde{u}_i = \emptyset$ and $vars(x_i\theta) \cap \tilde{u}_i = \emptyset$ to dissallow the environment to use restricted names and variables which are supposed to be secret; thus, they ensure that the value $x_i\theta$ can be *deduced* from public data. We denote by $Sol_{\mathcal{E}}(\mathcal{C})$ the set of $\mathcal{E}$-solutions of $\mathcal{C}$ and by $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C})$ the set of all closed $\mathcal{E}$-solutions of $\mathcal{C}$.

Example 5.10 *Let* $\mathcal{C} = \{\nu k.\nu s.\{{}^{\mathsf{enc}(s,k)}/_{y_1}, {}^{k}/_{y_2}\} \Vdash x', \mathsf{gd}(c), x' = s\}$. *Let* $\mathcal{E}$ *be the equational theory* $\mathsf{dec}(\mathsf{enc}(x,y),y) = x$ *and* $\theta = \{{}^{\mathsf{dec}(y_1,y_2)}/_{x'}\}$. *We have that* θ *is a closed* $\mathcal{E}$*-solution of* $\mathcal{C}$. *Note that* $\theta' = \{{}^{\mathsf{dec}(y_1,k)}/_{x'}\}$ *is not an* $\mathcal{E}$*-solution of* $\mathcal{C}$. *Indeed,* $names(x'\theta') \cap \{k,s\} = \{k\}$ *and thus is not empty.*

5.3.4 Symbolic semantics

A symbolic process is a triple $(A \,;\, \mathcal{C} \,;\, \mathsf{N}_s)$ where A is an intermediate extended process, $\mathcal{C}$ is a constraint system and N_s is a symbolic naming environment. A symbolic naming environment is similar to an intermediate naming environment, but may map variables to c for “constraint”. We obviously only consider well-formed processes such that A, $\mathcal{C}$ and N_s are consistent.

Given a well-formed symbolic process $(A \,;\, \mathcal{C} \,;\, \mathsf{N_s})$ we define by $Sol_{\mathcal{E}}(\mathcal{C} \,;\, \mathsf{N}_s)$, resp. $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C} \,;\, \mathsf{N}_s)$, the set of solutions, resp. closed solutions, of $\mathcal{C}$ which are compatible with N_s. In order to link symbolic and intermediate processes we note that each solution $\theta \in Sol_{\mathcal{E}}(\mathcal{C} \,;\, \mathsf{N_s})$ defines a corresponding (closed) intermediate process which we call the θ-concretization.

Definition 5.14 (θ-concretization) *Let* $(A_s \,;\, \mathcal{C} \,;\, \mathsf{N}_s)$ *be a well-formed symbolic process. Let* $\theta \in Sol_{\mathcal{E}}(\mathcal{C}, \mathsf{N}_s)$. *We say that an intermediate process* $(A \,;\, \mathsf{N})$ *is the* θ*-concretization of* $(A_s \,;\, \mathcal{C} \,;\, \mathsf{N}_s)$ *if* $A = A_s(\theta\sigma)^\star$ *where* σ *is the maximal frame of* $\mathcal{C}$ *and* $\mathsf{N} = \mathsf{N}_s|_{\mathcal{N}\cup\mathcal{X}}$.

We define the symbolic semantics by the means of the relations $\equiv_s$, $\rightarrow_s$ and $\stackrel{\alpha}{\rightarrow}_s$. Intuitively, when processes evolve they add constraints to $\mathcal{C}$. For instance, the input rule adds a deduction constraint and the conditionals add equality and disequality constraints. We show that each of these relations is sound and complete w.r.t. its intermediate counterpart.

In order to be able to define our symbolic labelled bisimulation we first define symbolic static equivalence using an encoding similar to the one in [Bau05]. The tests used to distinguish two frames in the definition of static equivalence are encoded by means of two additional deduction constraints on fresh variables x, y and by the equation $x = y$.

Definition 5.15 (symbolic static equivalence) *We say that two closed well-formed symbolic processes* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s)$ *and* $(B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$ *are symbolically statically equivalent, written* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s) \sim_s (B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$ *if for some variables* x, y*, the constraint systems* $\mathcal{C}'_A$, $\mathcal{C}'_B$ *have the same basis and* $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}'_A \,;\, \mathsf{N}'_s) = Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}'_B \,;\, \mathsf{N}'_s)$ *where*

- $\mathsf{N}_s(\{x, y\}) = \mathsf{n}$,
- $\mathsf{N}'_s = \mathsf{N}_s[x, y \mapsto \mathsf{c}]$,
- $\mathcal{C}'_A = \mathcal{C}_A \cup \{\phi(A_s) \Vdash x \;,\; \phi(A_s) \Vdash y \;,\; x = y\}$, *and*
- $\mathcal{C}'_B = \mathcal{C}_B \cup \{\phi(B_s) \Vdash x \;,\; \phi(B_s) \Vdash y \;,\; x = y\}$.

The following proposition states the correctness of the symbolic static equivalence with respect to the concrete one.

Proposition 5.2 (soundness of symbolic static equivalence) *Let* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s)$ *and* $(B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$ *be two closed and well-formed symbolic processes such that* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s) \sim_s (B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$*. Then we have that:*

1. $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}_A \,;\, \mathsf{N}_s) = Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}_B \,;\, \mathsf{N}_s)$, *and*
2. *for all* $\theta \in Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}_A \,;\, \mathsf{N}_s)$ *we have that* $\phi(A_s(\theta\sigma_A)^\star) \sim \phi(B_s(\theta\sigma_B)^\star)$*, where* σ_A *(resp.* σ_B*) is the substitution corresponding to the maximal frame of* $\mathcal{C}_A$ *(resp.* $\mathcal{C}_B$*).*

Although we do not need completeness of symbolic static equivalence for our result, we may note that it follows from Baudet's result [Bau05]. By completeness we mean that $A \sim B$ implies that $(A \,;\, \emptyset \,;\, \mathsf{N}_s) \sim_s (B \,;\, \emptyset \,;\, \mathsf{N}_s)$ for any compatible naming environment N_s.

We now define symbolic labelled bisimulation using our symbolic semantics.

Definition 5.16 (Symbolic labelled bisimilarity ($\approx_{\ell s}$)) *Symbolic labelled bisimilarity is the largest symmetric relation* $\mathcal{R}$ *on closed well-formed symbolic processes with same naming environment, such that* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s)\mathcal{R}(B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$ *implies*

1. $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s) \sim_s (B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s)$
2. *if* $(A_s \,;\, \mathcal{C}_A \,;\, \mathsf{N}_s) \rightarrow_s (A'_s \,;\, \mathcal{C}'_A \,;\, \mathsf{N}_s)$ *with* $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}'_A \,;\, \mathsf{N}_s) \neq \emptyset$*, then there exists a symbolic process* $(B'_s \,;\, \mathcal{C}'_B \,;\, \mathsf{N}_s)$ *such that*
   - $(B_s \,;\, \mathcal{C}_B \,;\, \mathsf{N}_s) \rightarrow^*_s (B'_s \,;\, \mathcal{C}'_B \,;\, \mathsf{N}_s)$, *and*

- $(A'_s\ ;\ \mathcal{C}'_A\ ;\ \mathsf{N_s})\ \mathcal{R}\ (B'_s\ ;\ \mathcal{C}'_B\ ;\ \mathsf{N}_s)$;

3. *if* $(A_s\ ;\ \mathcal{C}_A\ ;\ \mathsf{N}_s) \xrightarrow{\alpha_s}_s (A'_s\ ;\ \mathcal{C}'_A\ ;\ \mathsf{N}'_s)$ *with* $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}'_A\ ;\ \mathsf{N}'_s) \neq \emptyset$, *then there exists a symbolic process* $(B'_s\ ;\mathcal{C}'_B\ ;\ \mathsf{N}'_s)$ *such that*

- $(B_s\ ;\mathcal{C}_B\ ;\ \mathsf{N}_s) \rightarrow^*_s \xrightarrow{\alpha_s}_s \rightarrow^*_s (B'_s\ ;\mathcal{C}'_B\ ;\ \mathsf{N}'_s)$, *and*
- $(A'_s\ ;\ \mathcal{C}'_A\ ;\ \mathsf{N}'_s)\ \mathcal{R}\ (B'_s\ ;\mathcal{C}'_B\ ;\ \mathsf{N}'_s)$.

The side condition $Sol^{\mathsf{cl}}_{\mathcal{E}}(\mathcal{C}'_A\ ;\ \mathsf{N}'_s) \neq \emptyset$ ensures that we only consider symbolic executions that correspond to at least one concrete execution.

5.3.5 Soundness of symbolic bisimulation

We can now state our main result: the soundness of symbolic bisimulation.

Theorem 5.4 (Soundness of symbolic bisimulation) *Let A and B be two closed, nv-distinct extended processes. For any symbolic naming environment N_s compatible with $A{\downarrow}$, $B{\downarrow}$ and the empty constraint system we have that*

$$(A{\downarrow}\ ;\ \emptyset\ ;\ \mathsf{N}_s) \approx_{\ell s} (B{\downarrow}\ ;\ \emptyset\ ;\ \mathsf{N}_s) \text{ implies } A \approx_\ell B$$

Note that limiting the theorem to nv-distinct processes is not an onerous restriction. If we want to prove that $A \approx_\ell B$, we can construct by α-conversion two nv-distinct processes A', B' such that $A' \equiv A$ and $B' \equiv B$. Showing $A' \approx_\ell B'$ implies that $A \approx_\ell B$, since $\approx_\ell$ is closed under structural equivalence.

However, our symbolic bisimulation is not complete. Our techniques suffer from the same sources of incompleteness as the ones described for the spi calculus in [BBN04]. In a symbolic bisimulation the instantiation of input variables is postponed until the point at which they are actually used, leading to a finer relation.

Although our symbolic bisimulation is not complete we are able to prove labelled bisimulation on interesting examples for which the method implemented in the state-of-the-art ProVerif tool [Bla01] fails. For instance, ProVerif is unable to establish labelled bisimilarity between $\mathsf{out}(c,a) \mid \mathsf{out}(c,b)$ and $\mathsf{out}(c,b) \mid \mathsf{out}(c,a)$ whereas of course we are able to deal with such examples. A more interesting example, for which our symbolic semantics plays an important role is as follows.

Example 5.11 *Consider the following two processes.*

$$\begin{aligned} P &= \nu c_1.(\mathsf{in}(c_2,x).\mathsf{out}(c_1,x).\mathsf{out}(c_2,a) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,y)) \\ Q &= \nu c_1.(\mathsf{in}(c_2,x).\mathsf{out}(c_1,x).\mathsf{out}(c_2,x) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,a)) \end{aligned}$$

These two processes are labelled bisimilar and our symbolic labelled bisimulation is complete enough to prove this. In particular, let $P' = \nu c_1.(\mathsf{out}(c_1,x').\mathsf{out}(c_2,a) \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,y))$ *and* $Q' = \nu c_1.(\mathsf{out}(c_1,x').\mathsf{out}(c_2,x') \mid \mathsf{in}(c_1,y).\mathsf{out}(c_2,a))$. *The relation $\mathcal{R}$, that witnesses the symbolic bisimulation, includes*

$$\begin{array}{rcl} (P\ ;\ \emptyset\ ;\ \mathsf{N}_s) & \mathcal{R} & (Q\ ;\ \emptyset\ ;\ \mathsf{N}_s) \\ (P'\ ;\ \{\nu c_1.0 \Vdash x'\ ,\ \mathsf{gd}(c_2)\}\ ;\ \mathsf{N}'_s) & \mathcal{R} & (Q'\ ;\ \{\nu c_1.0 \Vdash x'\ ,\ \mathsf{gd}(c_2)\}\ ;\ \mathsf{N}'_s) \\ (\nu c_1.(\mathsf{out}(c_2,a) \mid \mathsf{out}(c_2,x'))\ ; & \mathcal{R} & (\nu c_1.(\mathsf{out}(c_2,x') \mid \mathsf{out}(c_2,a))\ ; \\ \{\nu c_1.0 \Vdash x'\ ,\ \mathsf{gd}(c_2)\ ,\ \mathsf{gd}(c_1)\}\ ;\ \mathsf{N}'_s) & & \{\nu c_1.0 \Vdash x'\ ,\ \mathsf{gd}(c_2)\ ,\ \mathsf{gd}(c_1)\}\ ;\ \mathsf{N}'_s) \end{array}$$

The example above is inspired by the problems we encountered when we analysed a bisimulation representing the privacy property in an electronic voting protocol. ProVerif is not able to prove this kind of equivalence; its algorithm is limited to cases that the two processes P, Q have the same structure, and differ only in the terms that are output. In this example, the processes differ in their structure, providing the motivation for our methods. Our symbolic bisimulation seems to be "sufficiently" complete to deal with examples of privacy and anonymity properties arising in protocol analysis. We demonstrate that more fully in an example in [9], which considers the privacy property of a simplified version of the FOO [FOO92] protocol.

We may note that recently sound and complete bisimulations have been proposed by Borgström [Bor08] for the spi calculus, by Johansson *et al.* [JVP09] for psi-calculi and by Liu and Lin for the applied pi calculus [LL10]. None of these bisimulations however directly yields a decision procedure. The latter work actually relies on our intermediate semantics. Cortier and Delaune [CD09a] have also shown that for determinate processes trace equivalence and observational equivalence coincide and adapted our work to yield a sound and complete symbolic bisimulation for this class of processes. Relying on the work by Baudet [Bau07] or more recent work by Cheval *et al.* [CCD10] or Chevalier and Rusinowitch [CR10] our work and the one by Cortier and Delaune provide decidability results for symbolic bisimulations for finite processes with no else branches when the equational theory is subterm convergent.

5.4 Conclusion and perspectives

In this chapter we summarized our results on the decision of equivalence properties. While the picture for static equivalence is rather complete and efficient tools exist for large classes of equational theories the situation is less clear when we consider indistinguishability in the presence of an active adversary. Even the definition of observational equivalence is questionable. An alternate, slightly weaker definition could be testing equivalence as introduced in the spi calculus. (Observational equivalence which has a characterization in terms of bisimulation can of course be used as a proof technique of testing equivalence.)

Decidability and complexity. Obviously, when we allow for replication, equivalence properties become undecidable. As shown by Hüttel [Hüt02], this is even the case for the finite control fragment, for which observational equivalence is decidable in the original pi calculus. Even though, one may expect observational and testing equivalence to be decidable for many classes of equational theories such results are currently missing and our theoretical understanding of this decision problem is currently very limited. Therefore we foresee to investigate decidability and complexity results for different (families of) equational theories.

One may note that equivalence of constraint systems such as [Bau07, CCD10, CR10], allows us to decide the equivalence of two symbolic traces. While this is an interesting first step towards the decision of equivalence properties it is not sufficient for deciding testing equivalence or observational equivalence (except for restricted classes of processes [CC10]). (These results nevertheless allow us to approximate observational equivalence using for instance our symbolic bisimulations.)

Combination. In order to develop decision procedure in a modular way it would be interesting to obtain combination results of the following type: if observational equivalence is decidable for the equational theory $\mathcal{E}_1$ and for the equational theory $\mathcal{E}_2$ then observational equivalence is also decidable for $\mathcal{E}_1 \cup \mathcal{E}_2$. Such results have been obtained for disjoint theories in the case of static equivalence [ACD07], as well as for disjoint theories [CR05] and hierarchical theories [CR06] in the case of reachability properties. To the best of our knowledge no such combination results exist for equivalence properties in the presence of an active attacker.

Efficient procedures. While a theoretical understanding of the decidability and complexity is important and interesting, it is equally important to develop practical tools that can be used to analyse and verify protocols. With Ş. Ciobâcă and R. Chadha we are currently working on a generalization of the procedure implemented in KiSs to decide trace equivalence for the finite applied pi calculus and selected equational theories.

CHAPTER 6

Modular analysis of security protocols

As protocols are getting more and more complex and are used as parts of large systems it is interesting to reason about them in a modular and compositional way. We would like to analyze protocols in isolation and conclude that they are secure in an arbitrary environment when executed in the presence of other protocols. As, by definition, security properties hold when executed in the presence of an arbitrary environment security is preserved when protocols do not share any secrets. Therefore, one might want to never use the same keying material for different protocols. This is however unrealistic in many situations. Distributing keys and getting them certified is often difficult and when smartcards are used for key management the available key-storage may limit the number of keys. We may also mention the notion of *cryptographic agility* [ABBC10] where different schemes (meeting a same security notion) reuse the same key and make key sharing between different protocols (or versions of a protocol) more likely.

In this vein, Cortier *et al.* [CDD07, CD09b] have investigated parallel composition of protocols which share some secrets. Their result can be summarized as

$$\textbf{if} \quad \nu k.P \models \varphi \quad \textbf{then} \quad \nu k.(P \mid Q) \models \varphi$$

for an arbitrary process Q provided that any two encrypted submessages coming from two different protocol specifications cannot be unified. This condition can easily be achieved using a *tagging* mechanism: in each encryption a tag, *e.g.*, the protocol name, is added to the plaintext before encrypting (and similar for other primitives). Their results hold for protocols using symmetric and asymmetric encryption, hash functions and signatures. The security property φ is expressed in a fragment of the logic PS-LTL (defined in [CES06]), covering essentially trace properties. Cortier and Ciobâcă [CC10] have recently generalized this composition result by allowing an arbitrary interleaving of two processes for any equational theory given that the two protocols use disjoint primitives. Again the disjointness of the primitives can be ensured for many primitives using a tagging mechanism. Their result allows in particular to achieve sequential composition, *e.g.*, show that a key exchange protocol establishing a shared key and a protocol supposing a shared key can be composed safely.

A situation where not reusing secrets is particular unrealistic arises when secrets are user-chosen passwords. Typically, a user will reuse the same password for different protocols. With S. Delaune and M. Ryan [27] we have studied the compositionality of resistance against offline dictionary attacks for password based protocols (a property which is not covered by previous composition results). We have shown that

if $\nu p.P$ is resistant against offline guessing attacks
and $\nu p.Q$ is resistant against offline guessing attacks
then $\nu p.(P \mid Q)$ is resistant against offline guessing attacks

holds in the case one considers a passive adversary, but not in general in the presence of an active adversary. We show that again a simple transformation, where the password is tagged by a protocol identifier using a hash function, both preserves resistance against offline dictionary attacks and guarantees composition even if the same password is reused in different protocols. Note that different ways of tagging may compromise resistance against guessing attacks, *i.e.*, the untagged protocol is secure while the tagged one is not. These results will be described in more detail in Section 6.1.

Another situation where the same keying material is reused arises when several sessions of a same protocol which uses long-term keys are executed. With M. Arapinis and S. Delaune [26], we have studied this kind of self-composition: the aim is to show that when a protocol is secure for one session it is also secure when composed with itself in many sessions, *i.e.*,

$$\textbf{if} \quad \nu k.P \models \varphi \quad \textbf{then} \quad \nu k.!P \models \varphi$$

While this is obviously not true in general we have designed a compiler which adds a preamble to the protocol where participants agree on a *dynamically* created session tag included in each encryption, signature and hash. For this class of compiled protocols we show that confidentiality properties are preserved under self-composition. We describe the result in more details in Section 6.2. This result can also be interpreted as a class of protocols for which confidentiality is decidable for an unbounded number of sessions. The use of tags to ease protocol analysis can also be found in several other works [BP03, RS03, RS05, AD07].

In computational models, universal composability (UC) [Can01] and reactive simulatability (RSIM) [BPW07] were designed to achieve composability. In UC and RSIM frameworks ideal functionalities, which one can think of as specifications, can be refined while preserving security protocols. With S. Delaune and O. Pereira [24] we have designed a similar framework for the applied pi calculus. While at first we thought that this would be a straightforward exercise, with the aim of better understanding the corresponding computational models, it turned out to be a non trivial task. In particular the concurrency model of the applied pi calculus differs significantly from the sequential scheduling mechanisms in computational models. Changing the concurrency model raised several interesting questions and led to different design choices. We detail this work in Section 6.3. We also note that in order to obtain joint state results, which correspond to composition with shared secrets, a unique session identifier is supposed to be given. While it is often omitted how such a session id can be established one way would be to use the technique we propose in [26] to establish session tags. These session ids are then used in a similar way as the tags in other symbolic composition results.

Please note that the above discussion on composition is not exhaustive and we omit for instance to discuss different frameworks for compositional reasoning such as the PCL logic [DDMR07] or composition results in other symbolic models, e.g., results in the strand space model [GT00c]. Also the use of tags, which shows to be one of the main tools to achieve composition, was already hinted to as a prudent engineering rule [AN06]. We here go one step further showing that we can actually prove the soundness of some "safe" engineering rules.

6.1 Composition of password-based protocols

6.1.1 Modelling guessing attacks

We first define what it means for a frame to be resistant against offline guessing attacks. The idea behind the definition is the following. Suppose the frame ϕ represents the information gained by the attacker by eavesdropping one or more sessions and let w be the weak password. Then, we can represent resistance against guessing attacks by checking whether the attacker can distinguish a situation in which he guesses the correct password w and a situation in which he guesses an incorrect one, say w'. We model these two situations by adding $\{^{w}/_{x}\}$ (resp. $\{^{w'}/_{x}\}$) to the frame. We use static equivalence to capture the notion of indistinguishability. This definition is due to Baudet [Bau05], inspired from the one of [CDE05]. In our definition, we allow multiple shared secrets, and write $\tilde{w}$ for a sequence of such secrets.

Definition 6.1 *Let $\phi \equiv \nu\tilde{w}.\phi'$ be a frame. We say that the frame ϕ is* resistant to guessing attacks *against $\tilde{w}$ if*

$$\nu\tilde{w}.(\phi' \mid \{^{\tilde{w}}/_{\tilde{x}}\}) \sim \nu\tilde{w}.(\phi' \mid \nu\tilde{w}'.\{^{\tilde{w}'}/_{\tilde{x}}\})$$

where $\tilde{x}$ a sequence of variables such that $\tilde{x} \cap \mathsf{dom}(\phi) = \emptyset$.

Note that this definition is general w.r.t. to the equational theory and the number of guessable data items.

Now, we can define what it means for a protocol to be resistant against guessing attacks (in presence of an active attacker). Intuitively, a protocol A is resistant against guessing attacks on a weak password w if it is not possible for an active attacker to mount a guessing attack on it even after some interactions with the protocol during a first phase. In other words, for any process B such that $A \Longrightarrow B$ (note that the attacker can intercept and send messages during this phase), the frame $\phi(B)$ has to be resistant to guessing attack.

Definition 6.2 *Let A be a process and $\tilde{w} \subseteq bn(A)$. We say that A is* resistant to guessing attacks *against $\tilde{w}$ if, for every process B such that $A \Longrightarrow B$, we have that the frame $\phi(B)$ is resistant to guessing attacks against $\tilde{w}$.*

Example 6.1 *In the remaining of this section we will consider the equational theory $\mathcal{E}_{\mathsf{enc}}$, defined by the following equations:*

$$\begin{array}{rclcrcll} \mathsf{sdec}(\mathsf{senc}(x,y),y) & = & x & \qquad & \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) & = & x & \\ \mathsf{senc}(\mathsf{sdec}(x,y),y) & = & x & & \mathsf{proj}_i(\langle x_1,x_2\rangle) & = & x_i & (i \in \{1,2\}) \end{array}$$

Consider the following simple handshake protocol $\nu w.(A \mid B)$ *where*

- $A = \nu n.\mathsf{out}(c, \mathsf{senc}(n, w)).\ \mathsf{in}(c, x).$ *if* $\mathsf{sdec}(x, w) = f(n)$ *then* P
- $B = \mathsf{in}(y).\ \mathsf{out}(\mathsf{senc}(f(\mathsf{sdec}(y, w)), w)).$

where P *models an application that is executed when* B *has been successfully authenticated. An interesting problem arises if the shared key* w *is a weak secret, i.e. vulnerable to brute-force off-line testing. In such a case, the protocol has a guessing attack against* w*. Indeed, we have that* $\nu w.(A \mid B) \Longrightarrow D$ *with* $\phi(D) = \nu w.\nu n.(\{^{\mathsf{senc}(n,w)}/_{x_1}\} \mid \{^{M}/_{x_2}\})$.

The frame $\phi(D)$ *is not resistant to guessing attacks against* w*. The test* $f(\mathsf{sdec}(x_1, x)) \stackrel{?}{=} \mathsf{sdec}(x_2, x)$ *allows us to distinguish the two associated frames:*

- $\nu w.\nu n.(\{^{\mathsf{senc}(n,w)}/_{x_1}\} \mid \{^{M}/_{x_2}\} \mid \{^{w}/_{x}\})$*, and*
- $\nu w'.\nu w.\nu n.(\{^{\mathsf{senc}(n,w)}/_{x_1}\} \mid \{^{M}/_{x_2}\} \mid \{^{w'}/_{x}\})$.

This corresponds to the classical guessing attack on the handshake protocol (see [Gon93]). After a normal execution of one session of this protocol, the attacker learns two messages, namely $\mathsf{senc}(n, w)$ *and* $\mathsf{senc}(f(n), w)$*. By decrypting these two messages with his guess* x*, he can easily test whether* $x = w$ *and thus recover the weak password* w *by brute-force testing.*

6.1.2 Composition Result – Passive Case

The goal of this section is to establish a composition result in the passive case for resistance against guessing attacks. We first show the equivalence of three definitions of resistance against guessing attacks: the first definition is due to Baudet [Bau05] and the second one is due to Corin *et al.* [CDE05]. The last definition is given in a composable way and establishes our composition result (see Corollary 6.1).

Proposition 6.1 *Let* ϕ *be a frame such that* $\phi \equiv \nu\tilde{w}.\phi'$*. The three following statements are equivalent:*

1. ϕ *is resistant to guessing attacks against* $\tilde{w}$ *(according to Definition 6.1),*
2. $\phi' \sim \nu\tilde{w}.\phi'$*,*
3. $\phi' \sim \phi'\{^{\tilde{w}'}/_{\tilde{w}}\}$ *where* $\tilde{w}'$ *is a sequence of fresh names.*

Now, by relying on Proposition 6.1 (item *3.*), it is easy to show that resistance to guessing attack against $\tilde{w}$ for two frames that share only the names $\tilde{w}$ is a composable notion. This is formally stated in the corollary below:

Corollary 6.1 *Let* $\phi_1 \equiv \nu\tilde{w}.\phi_1'$ *and* $\phi_2 \equiv \nu\tilde{w}.\phi_2'$ *be two frames. If* ϕ_1 *and* ϕ_2 *are resistant to guessing attacks against* $\tilde{w}$ *then* $\nu\tilde{w}.(\phi_1' \mid \phi_2')$ *is also resistant to guessing attacks against* $\tilde{w}$*.*

Note that a similar result does not hold for deducibility: even if w is neither deducible from ϕ_1 nor from ϕ_2, it can be deducible from $\phi_1 \mid \phi_2$.

$$
\begin{array}{lll}
A & = & \nu k, na. \\
 & & \mathsf{out}(\mathsf{senc}(\mathsf{pk}(k), w)). \\
 & & \mathsf{in}(x_1). \\
 & & \text{let } ra = \mathsf{adec}(\mathsf{sdec}(x_1, w), k). \\
 & & \mathsf{out}(\mathsf{senc}(na, ra)). \\
 & & \mathsf{in}(x_2). \\
 & & \text{if } \mathsf{proj}_1(\mathsf{sdec}(x_2, ra)) = na \text{ then} \\
 & & \mathsf{out}(\mathsf{sdec}(\mathsf{proj}_2(\mathsf{sdec}(x_2, ra)), ra)).\, 0
\end{array}
\qquad
\begin{array}{lll}
B & = & \nu r, nb. \\
 & & \mathsf{in}(y_1). \\
 & & \mathsf{out}(\mathsf{senc}(\mathsf{aenc}(r, \mathsf{sdec}(y_1, w)), w)). \\
 & & \mathsf{in}(y_2). \\
 & & \mathsf{out}(\mathsf{senc}(\langle \mathsf{sdec}(y_2, r), nb \rangle, r)). \\
 & & \mathsf{in}(y_3). \\
 & & \text{if } \mathsf{sdec}(y_3, r) = nb \text{ then} \\
 & & 0
\end{array}
$$

Figure 6.1: Modelling of the EKE protocol

Example 6.2 *Consider again the equational theory* $\mathcal{E}_{\mathsf{enc}}$. *Consider the frames:* $\phi_1 = \{^{\mathsf{senc}(w,\mathsf{senc}(w,w))}/_{x_1}\}$ *and* $\phi_2 = \{^{\mathsf{senc}(w,w)}/_{x_2}\}$. *We have that* $\nu w.\phi_i \not\vdash_{\mathcal{E}} w$ *for* $i = 1, 2$ *whereas* $\nu w.(\{^{\mathsf{senc}(w,\mathsf{senc}(w,w))}/_{x_1}\} \mid \{^{\mathsf{senc}(w,w)}/_{x_2}\}) \vdash_{\mathcal{E}} w$. *Indeed, the term* $\mathsf{sdec}(x_1, x_2)$ *is a recipe of the term* w.

In the case of *password-only* protocols, i.e., protocols that only share a password between different sessions and do not have any other long-term shared secrets we have the following direct consequence. We can prove resistance against guessing attacks for an unbounded number of parallel sessions by proving only resistance against guessing attacks for a single session. An example of a password-only protocol is the well-known EKE protocol [BM92]. This protocol has also been analysed using the applied pi calculus in [CDE05] and in computationally sound settings in [AW05, ABW10].

Example 6.3 *The EKE protocol [BM92] can be informally described by the following 5 steps. A formal description of this protocol in our calculus is given in Figure 6.1.*

$\mathrm{A} \to \mathrm{B}:$	$\mathsf{senc}(\mathsf{pk}(k), w)$	*(EKE.1)*
$\mathrm{B} \to \mathrm{A}$	$\mathsf{senc}(\mathsf{aenc}(r, \mathsf{pk}(k)), w)$	*(EKE.2)*
$\mathrm{A} \to \mathrm{B}$	$\mathsf{senc}(na, r)$	*(EKE.3)*
$\mathrm{B} \to \mathrm{A}$	$\mathsf{senc}(\langle na, nb \rangle, r)$	*(EKE.4)*
$\mathrm{A} \to \mathrm{B}$	$\mathsf{senc}(nb, r)$	*(EKE.5)*

In the first step (EKE.1) A *generates a new private key* k *and sends the corresponding public key* $\mathsf{pk}(k)$ *to* B*, encrypted (using symmetric encryption) with the shared password* w. *Then,* B *generates a fresh session key* r*, which he encrypts (using asymmetric encryption) with the previously received public key* $\mathsf{pk}(k)$. *Finally, he encrypts the resulting ciphertext with the password* w *and sends the result to* A *(EKE.2). The last three steps (EKE.3-5) perform a handshake to avoid replay attacks. One may note that this is a password-only protocol. A new private and public key are used for each session and the only shared secret between different sessions is the password* w.

We use again the equational theory $\mathcal{E}_{\mathsf{enc}}$ *(cf Example 6.1) to model this protocol. An execution of this protocol in the presence of a passive attacker yields the frame* $\nu w.\phi$ *where*

$$\phi = \nu k, r, na, nb.\{ \begin{array}{l} {}^{\mathsf{senc}(\mathsf{pk}(k),w)}/_{x_1}, {}^{\mathsf{senc}(\mathsf{aenc}(r,\mathsf{pk}(k)),w)}/_{x_2}, \\ {}^{\mathsf{senc}(na,r)}/_{x_3}, {}^{\mathsf{senc}(\langle na,nb\rangle,r)}/_{x_4}, {}^{\mathsf{senc}(nb,r)}/_{x_5}\} \end{array}$$

We have that $\nu w.(\phi \mid \{^{w}/_{x}\}) \sim \nu w, w'.(\phi \mid \{^{w'}/_{x}\})$. *We have verified this static equivalence using the YAPA tool [Bau08].*

Corin *et al.* [CDE05] also analysed one session of this protocol (with a slight difference in the modelling). It directly follows from our previous result that the protocol is secure for any number of sessions as the only secret shared between different sessions is the password w.

6.1.3 Composition Result – Active Case

In the active case, contrary to the passive case, resistance against guessing attacks does not compose: even if two protocols separately resist against guessing attacks on w, their parallel composition under the shared password w may be insecure. Consider the following example.

Example 6.4 *Consider the processes defined in Figure 6.1 where the occurrence of* 0 *in* B *has been replaced by* $\mathsf{out}(w)$. *Let* A' *and* B' *be these two processes. The process* $\nu w.(A' \mid B')$ *models a variant of the EKE protocol where* B' *outputs the password* w *if the authentication of* A' *succeeds. We have that* $\nu w.A'$ *and* $\nu w.B'$ *resist against guessing attacks on* w. *We have verified these statements by using the ProVerif tool [Bla04]. However,* $\nu w.(A' \mid B')$ *trivially leaks* w. *More generally any secure password only authentication protocol can be modified in this way to illustrate that resistance against guessing attacks does not compose in the active case.*

The previous example may not be entirely convincing, since there is no environment in which either of the separate processes $\nu w.A'$ and $\nu w.B'$ is *executable*. We do not give a formal definition of what it means for a process to be executable. Therefore we present a second example (more complicated but in the same spirit) in which each of the constituent processes admits a complete execution.

Example 6.5 *Consider the processes* A *and* B *defined in Figure 6.1 where the occurrences of* 0 *in* A *and* B *have been replaced by* $\mathsf{out}(\mathsf{senc}(w, ra))$ *and* $\mathsf{in}(y).\mathsf{out}(\mathsf{sdec}(y, r))$ *respectively. Let* A_1 *and* B_1 *be these two processes. We can see* $\nu w.(A_1 \mid B)$ *and* $\nu w.(A \mid B_1)$ *as two extensions of the EKE protocol with an additional exchange. Note also that these two protocols admit a normal execution and in this sense are* executable. *We have that* $\nu w.(A_1 \mid B)$ *and* $\nu w.(A \mid B_1)$ *are resistant against guessing attacks on* w. *In particular the additional exchange does not lead to an attack. We have verified these statements using the tool ProVerif. However,* $\nu w.(A_1 \mid B_1)$ *and thus* $\nu w.((A_1 \mid B) \mid (A \mid B_1))$, *trivially leaks* w.

This example shows that there is no hope to obtain a general composition result that holds even for a particular and relatively simple equational theory. To reach our goal, we consider a restricted class of protocols: the class of *well-tagged* protocols.

6.1.4 Well-tagged Protocols

Intuitively, a protocol is well-tagged w.r.t. a secret w if all the occurrences of w are of the form $\mathsf{h}(\alpha, w)$. We require that h is a hash function (i.e., has no equations in the equational

theory), and α is a name, which we call the *tag*. The idea is that if each protocol is tagged with a different name (e.g. the name of the protocol) then the protocols compose safely. In the remainder, we will consider an arbitrary equational theory $\mathcal{E}$, provided there is no equation for h.

Definition 6.3 (well-tagged) *Let M be a term and w be a name. We say that M is* α-tagged *w.r.t. w if there exists M' such that $M'\{^{\mathsf{h}(\alpha,w)}/_{w}\} =_{\mathcal{E}} M$.*

A term is said well-tagged *w.r.t. w if it is α-tagged w.r.t. w for some name α. An extended process A is* α-tagged *if any term occurring in it is α-tagged. An extended process is* well-tagged *if it is α-tagged for some name α.*

A protocol can be easily transformed into a well-tagged protocol. We have shown that the simple transformation where we replace the password w with $\mathsf{h}(\alpha, w)$ indeed preserves resistance against offline guessing attacks.

Theorem 6.1 *Let $A \equiv \nu w.A'$ be a process resistant to guessing attacks against w, then we have that $\nu w.(A'\{^{\mathsf{h}(\alpha,w)}/_{w}\})$ is also resistant to guessing attacks against w.*

Other ways of tagging a protocol exist in the literature. For example, in [CDD07] encryption are tagged to ensure that they cannot be used to attack other protocols. That particular method would not work here; on the contrary, that kind of tagging is likely to add guessing attacks.

Example 6.6 *Let $A = \nu w, s.\mathsf{out}(\mathsf{senc}(s, w))$. We have that A is resistant to guessing attacks against w. However, the corresponding well-tagged protocol, according to the definition given in [CDD07], is not. Indeed,*

$$A' = \nu w, s.\mathsf{out}(\mathsf{senc}(\langle \alpha, s\rangle, w))$$

is not resistant to guessing attack against w. The tag α which is publicly known can be used to mount such an attack. An attacker can decrypt the message $\mathsf{senc}(\langle \alpha, s, \rangle, w)$ with his guess x and check whether the first component of the pair is the publicly known value α. Hence, he can test whether $x = w$ and recover the password w by brute force testing.

Another tagging method we considered is to replace w by $\langle \alpha, w\rangle$ (instead of $\mathsf{h}(\alpha, w)$), which has the advantage of being computationally cheaper. This transformation does not work, although the only counterexamples we have are rather contrived.

Example 6.7 *Consider the equational theory $\mathcal{E}_{\mathsf{enc}}$.*
Let $A = \nu w, k.\mathsf{out}(\mathsf{senc}(w, k)).in(x).$ if $\mathsf{proj}_1(\mathsf{dec}(x, k)) = \alpha$ then $\mathsf{out}(w)$.

Note that we can built a similar example without using α in the specification of A. We can simply compare the first component of two ciphertexts issued from the protocols. This should lead to an equality (i.e. a test) which does not necessarily exist in the original protocol. In [Syv96], Syverson already noted that *explicitness*, as recommended in prudent engineering principles [AN06, AN95], may break resistance against guessing attacks.

6.1.5 Composition Theorem

We show that any two well-tagged protocols that are separately resistant to guessing attacks can be safely composed provided that they use different tags. The following theorem formalizes the intuition that replacing the shared password with a hash parametrized by the password and a tag is similar to using different passwords which implies composition.

Theorem 6.2 (composition result) *Let A_1 and A_2 be two well-tagged processes w.r.t. w such that the process A_1 (resp. A_2) is α-tagged (resp. β-tagged) and $\nu w.(A_1 \mid A_2)$ is a process (this can be achieved by using α-renaming).*

If $\nu w.A_1$ and $\nu w.A_2$ are resistant to guessing attacks against w and $\alpha \neq \beta$, then we have that $\nu w.(A_1 \mid A_2)$ is also resistant to guessing attacks against w.

6.2 Self-composition using dynamic tags

We will describe this work only informally. In [26], the results are obtained in a role based model with pattern matching and a fixed set of cryptographic primitives (pairing, symmetric and public key encryption, signatures and hash functions) which we will not introduce here formally.

We consider the secrecy property and we propose a protocol transformation which maps a protocol that is secure for a single session to a protocol that is secure for an unbounded number of sessions. This provides an effective strategy to design secure protocols: (i) design a protocol intended to be secure for one protocol session (this can be efficiently verified with existing automated tools); (ii) apply our transformation and obtain a protocol which is secure for an unbounded number of sessions.

6.2.1 The protocol transformation

Given an input protocol Π, our transformation will compute a new protocol $\tilde{\Pi}$ which consists of two phases. During the first phase, the protocol participants try to agree on some common, dynamically generated, session identifier τ. For this, each participant sends a freshly generated nonce N_i together with his identity A_i to all other participants. (Note that if broadcast is not practical or if not all identities are known to each participant, the message can be sent to some of the participants who forward the message.) At the end of this preamble, each participant computes a session identifier: $\tau = \langle\langle A_1, N_1\rangle, \ldots, \langle A_k, N_k\rangle\rangle$. An active attacker may of course interfere with this initialization phase, intercept and replace some of the nonces. Hence, the protocol participants do not necessarily agree on the same session identifier τ after this preamble. In fact, each participant computes his own session identifier, say τ_j. During the second phase, each participant j executes the original protocol in which the dynamically computed identifier is used for tagging each application of a cryptographic primitive. In this phase, when a participant opens an encryption, he will check that the tag is in accordance with the nonces he received during the initialization phase. In particular he can test the presence of his own nonce.

The transformation, using the informal *Alice-Bob notation*, is described below and relies on the tagging operation $[m]_\tau$ which adds the tag τ to each application of a cryptographic primitive.

$$\Pi = \begin{cases} A_{i_1} \to A_{j_1} : & m_1 \\ \vdots & \\ A_{i_\ell} \to A_{j_\ell} : & m_\ell \end{cases} \qquad \tilde{\Pi} = \begin{cases} \text{Phase 1} & & \text{Phase 2} \\ A_1 \to All : & \langle A_1, N_1 \rangle & A_{i_1} \to A_{j_1} : & [m_1]_\tau \\ \vdots & & \vdots & \\ A_k \to All : & \langle A_k, N_k \rangle & A_{i_\ell} \to A_{j_\ell} : & [m_\ell]_\tau \\ \text{where } \tau = \langle \mathsf{tag}_1, \ldots, \mathsf{tag}_k \rangle \text{ with } \mathsf{tag}_i = \langle A_i, N_i \rangle \end{cases}$$

Note that the above Alice-Bob notation only represents what happens in a normal execution, i.e. with no intervention of the attacker. Of course, in such a situation, the participants agree on the same session identifier τ used in the second phase.

6.2.2 The composition result

In [26, Theorem 1], we show the following result, restated here.

> Let Π be a k-party protocol, $\tilde{\Pi}$ be its corresponding transformed protocol, and T_0 be a finite set of ground atoms (the adversary's initial knowledge).
>
> Let m be a term appearing in $\Pi(j)$ for some $1 \leq j \leq k$ and $\tilde{m}$ be its counterpart in $\tilde{\Pi}(j)$. Let CK, the critical keys, be the set of longterm keys not in T_0 and assume that keys in CK do not appear in plaintext.
>
> If Π preserves the secrecy of m w.r.t. T_0 when considering one honest session of each role, then $\tilde{\Pi}$ preserves the secrecy of $\tilde{m}$ w.r.t. T_0 (for an unbounded number of sessions).

Our result states that if the compiled protocol admits an attack that may involve several honest and dishonest sessions, then there exists an attack which only requires one honest session of each role (and no dishonest sessions). We may note that, *en passant*, we identify a class of tagged protocols for which security is decidable for an unbounded number of sessions. This directly follows from our main result as it stipulates that verifying security for a single protocol session is sufficient to conclude security for an unbounded number of sessions.

Our dynamic tagging is useful to avoid interaction between different sessions of the same role in a protocol execution and allows us for instance to prevent man-in-the-middle attacks. We have also considered an alternative, slightly different transformation that does not include the identities in the tag, i.e., the tag is simply the sequence of nonces. In that case we obtain a different result: if a protocol admits an attack then there exists an attack which only requires one (not necessarily honest) session for each role. In this case, we need to additionally check for attacks that involve a session engaged with the attacker. On the example of the Needham-Schroeder protocol the man-in-the-middle attack is not prevented by this weaker tagging scheme. However, the result requires one to also consider one dishonest session for each role, hence including the attack scenario. In both cases, it is important for the tags to be *collaborative*, *i.e.* all participants do contribute by adding a fresh nonce.

Different kinds of tags have also been considered in [AD07, BP03, RS03]. However these tags are *static* and have a different aim. While our dynamic tagging scheme avoids confusing messages from different sessions, these static tags avoid confusing different messages

inside a same session and do not prevent that a same message is reused in two different sessions. Under some additional assumptions (e.g. no temporary secret, no ciphertext forwarding), several decidability results [RS05, Low99] have been obtained by showing that it is sufficient to consider one session per role. But those results can not deal with protocols such as the Yahalom protocol or protocols which rely on a temporary secret. In the framework we consider here, the question whether such static tags would be sufficient to obtain decidability is still an open question (see [AD07]).

6.3 Simulation based security

In this section we present our symbolic simulation based framework.

6.3.1 Basic definitions

The simulation-based security approach classically distinguishes between *input-output channels*, which yield the internal interface of a protocol or *functionality* to its environment and *network channels*, which allow the adversary to interact with the functionality. We suppose that all channels are of one of these two sorts: IO or NET. Moreover the sort system ensures that names of sort NET can never be conveyed as data on a channel, i.e. these channels can never be transmitted. We write $\mathsf{fnet}(P)$ for $fn(P) \cap \mathsf{NET}$.

Definition 6.4 *A* functionality $\mathcal{F}$ *is a closed plain process. An* adversary *for* $\mathcal{F}$ *is an evaluation context* $\mathcal{A}[\_]$ *of the form:* $\nu\widetilde{net}_1.(A_1 \mid \nu\widetilde{net}_2.(A_2 \mid \ldots \mid \nu\widetilde{net}_k.(A_k \mid \_\,)\ldots))$ *where each* A_i $(1 \leq i \leq k)$ *is a closed plain process,* $\mathsf{fnet}(\mathcal{F}) \subseteq \bigcup_{1\leq i\leq k} \widetilde{net}_i \subseteq \mathsf{NET}$*, and* $fn(\mathcal{A}[\_]) \cap \mathsf{IO} = \emptyset$.

One may note that the nested form of the adversary process allows to express any arbitrary context while expliciting the restricted names whose scope ranges on the hole. We also note that if $\mathcal{A}[\_]$ is an adversary for $\mathcal{F}$ then $\mathsf{fnet}(\mathcal{A}[\_]) = \mathsf{fnet}(\mathcal{A}[\mathcal{F}])$.

Lemma 6.1 *Let* $\mathcal{F}$ *be a functionality and* $\mathcal{A}_1[\_]$ *be an adversary for* $\mathcal{F}$*. Then* $\mathcal{A}_1[\mathcal{F}]$ *is a functionality. If* $\mathcal{A}_2[\_]$ *is an adversary for* $\mathcal{A}_1[\mathcal{F}]$*, then* $\mathcal{A}_2[\mathcal{A}_1[\_]]$ *is an adversary for* $\mathcal{F}$.

While adversaries control the communication of functionalities on NET channels, IO *contexts* model the environment of functionalities, providing inputs and collecting outputs.

Definition 6.5 *An* IO context *is an evaluation context* $C_{io}[\_]$ *of the form* $\nu\widetilde{io}_1.(C_1 \mid \nu\widetilde{io}_2.(C_2 \mid \ldots \mid \nu\widetilde{io}_k.(C_k \mid \_\,)\ldots))$ *where each* C_i $(1 \leq i \leq k)$ *is a closed plain process, and* $\bigcup_{1\leq i\leq k} \widetilde{io}_i \subseteq \mathsf{IO}$

Note that if $\mathcal{F}$ is a functionality and $C_{io}[\_]$ is an IO context, then $C_{io}[\mathcal{F}]$ is a functionality.

6.3.2 Strong simulatability

The notion of strong simulatability [KDMR08], which is probably the simplest secure emulation notion used in simulation-based security, can be formulated in our setting.

Definition 6.6 (strong simulatability) *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two functionalities. $\mathcal{F}_1$* emulates *$\mathcal{F}_2$ in the sense of* strong simulatability*, written $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$, if there exists an adversary $\mathcal{S}$ for $\mathcal{F}_2$ (the simulator) such that $\mathsf{fnet}(\mathcal{F}_1) = \mathsf{fnet}(\mathcal{S}[\mathcal{F}_2])$ and $\mathcal{F}_1 \preceq \mathcal{S}[\mathcal{F}_2]$.*

The definition ensures that any behavior of $\mathcal{F}_1$ can be matched by $\mathcal{F}_2$ executed in the presence of a specific adversary $\mathcal{S}$. Hence, there are no more attacks on $\mathcal{F}_1$ than attacks on $\mathcal{F}_2$. Moreover, the presence of $\mathcal{S}$ allows abstract definitions of higher-level functionalities, which are independent of a specific realization. One may also note that $0 \leq^{\mathsf{SS}} \mathcal{F}$ for any functionality $\mathcal{F}$. This seems natural in a simulation based framework which only aims at preserving security. Non-triviality conditions may be imposed independently [Can01].

Example 6.8 *Let*

$$\mathcal{F}_{\mathsf{cc}} = \mathsf{in}(io_1, x_s).\mathsf{out}(net_{\mathsf{cc}}, \mathsf{ok}).\mathsf{in}(net_{\mathsf{cc}}, x).\ \textit{if } x = \mathsf{ok} \textit{ then } \mathsf{out}(io_2, x_s)$$

The functionality models a confidential channel and takes a potentially secret value s as input on channel io_1. The adversary is notified via channel net_{cc} that this value is to be transmitted. If the adversary agrees the value is output on channel io_2. This functionality can be realized by the process

$$\begin{aligned} P = \nu k.\,(\quad & \mathsf{in}(io_1, x).\mathsf{out}(net, \mathsf{senc}(x,k)) \mid \\ & \mathsf{in}(net, y).\ \textit{if } \mathsf{test}(y,k) = \mathsf{ok} \textit{ then } \mathsf{out}(io_2, \mathsf{sdec}(y,k)) \textit{ else } 0) \end{aligned}$$

assuming the equational theory induced by

$$\mathsf{sdec}(\mathsf{senc}(x,k),k) = x \qquad \mathsf{test}(\mathsf{senc}(x,y),y) = \mathsf{ok}$$

Let

$$\mathcal{S} = \nu net_{\mathsf{cc}}.(\mathsf{in}(net_{\mathsf{cc}}, x).\nu r.\mathsf{out}(net, r) \mid \mathsf{in}(net, x).\ \textit{if } x = r \textit{ then } \mathsf{out}(net_{\mathsf{cc}}, \mathsf{ok}) \mid \_\,)$$

We indeed have that $P \preceq_\ell \mathcal{S}[\mathcal{F}_{\mathsf{cc}}]$ and $\mathsf{fnet}(P) = \mathsf{fnet}(\mathcal{S}[\mathcal{F}_{\mathsf{cc}}])$.

In order to examine the properties of strong simulatability in our specific setting, we now define a particular adversary $D[\_]$ which is called a *dummy adversary*: intuitively, it acts as a relay which forwards all messages. The formal definition is technical because $D[\_]$ needs to both restrict all names in $\mathsf{fnet}(\mathcal{F})$ and ensure that $\mathsf{fnet}(\mathcal{F}) = \mathsf{fnet}(D[\mathcal{F}])$. It therefore relies on two internal channels $sim_j^{i/o}$ for inputs, resp. outputs, for each channel in $\mathsf{fnet}(\mathcal{F})$.

Definition 6.7 (dummy adversary) *Let $\mathcal{F}$ be a functionality. The* dummy adversary *for $\mathcal{F}$ is the adversary $D[\_] = \nu\widetilde{sim}.(D_1 \mid \nu\widetilde{net}.(D_2 \mid \_\,))$ where $\widetilde{net} = \mathsf{fnet}(\mathcal{F}) = \{net_1, \ldots, net_n\}$; $\widetilde{sim} = \{sim_1^i, \ldots, sim_n^i, sim_1^o, \ldots, sim_n^o\} \subseteq \mathsf{NET}$; and*

- $D_1 = \begin{aligned}[t] & !\mathsf{in}(net_1, x).\mathsf{out}(sim_1^i, x) \mid \ldots \mid !\mathsf{in}(net_n, x).\mathsf{out}(sim_n^i, x) \mid \\ & !\mathsf{in}(sim_1^o, x).\mathsf{out}(net_1, x) \mid \ldots \mid !\mathsf{in}(sim_n^o, x).\mathsf{out}(net_n, x); \end{aligned}$
- $D_2 = \begin{aligned}[t] & !\mathsf{in}(sim_1^i, x).\mathsf{out}(net_1, x) \mid \ldots \mid !\mathsf{in}(sim_n^i, x).\mathsf{out}(net_n, x) \mid \\ & !\mathsf{in}(net_1, x).\mathsf{out}(sim_1^o, x) \mid \ldots \mid !\mathsf{in}(net_n, x).\mathsf{out}(sim_n^o, x). \end{aligned}$

By construction we have that $\mathsf{fnet}(D[\mathcal{F}]) = \mathsf{fnet}(\mathcal{F})$.

Lemma 6.2 *Let $\mathcal{F}$ be a functionality and $D[\_]$ be the dummy adversary for $\mathcal{F}$: $\mathcal{F} \preceq D[\mathcal{F}]$.*

However, we do not have that $\mathcal{F} \approx D[\mathcal{F}]$, since $D[\mathcal{F}]$ has more non-determinism than $\mathcal{F}$. A direct consequence of this lemma is that $\mathcal{F}_1 \preceq \mathcal{F}_2$ and $\mathsf{fnet}(\mathcal{F}_1) = \mathsf{fnet}(\mathcal{F}_2)$ implies that $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$: as $\mathcal{F}_2 \preceq D[\mathcal{F}_2]$ we have by transitivity that $\mathcal{F}_1 \preceq D[\mathcal{F}_2]$. We use these observations to show that $\leq^{\mathsf{SS}}$ is a preorder (Lemma 6.3) , which is closed under application of IO contexts (Proposition 6.2) and parallel composition (Proposition 6.3).

Lemma 6.3 *Let $\mathcal{F}_1$, $\mathcal{F}_2$ be two functionalities. We have that:*

1. $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_1$;
2. $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *and* $\mathcal{F}_2 \leq^{\mathsf{SS}} \mathcal{F}_3 \;\Rightarrow\; \mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_3$.

Proposition 6.2 *Let $\mathcal{F}_1$, $\mathcal{F}_2$ be two functionalities and C_{io} be an IO context.*

$$\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2 \implies C_{io}[\mathcal{F}_1] \leq^{\mathsf{SS}} C_{io}[\mathcal{F}_2].$$

Proposition 6.3 *Let $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ be three functionalities. We have that:*

1. $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2 \;\Rightarrow\; \mathcal{F}_1 \mid \mathcal{F}_3 \leq^{\mathsf{SS}} \mathcal{F}_2 \mid \mathcal{F}_3$; *and*
2. $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2 \;\Rightarrow !\mathcal{F}_1 \leq^{\mathsf{SS}} !\mathcal{F}_2$.

While, (1) is a direct consequence of Proposition 6.2 (notice that $\_ \mid \mathcal{F}_3$ is an IO-context) the proof of (2) is more involved.

Remark 6.1 *The idea of comparing a security protocol to an idealized version of this protocol which is correct by construction goes back to [GMW87]. In a symbol setting the use of observational equivalence for this purpose was presented in the spi calculus [AG99]. In [AG99] the protocol is expected to be observationally equivalent to an ideal functionality without the presence of a simulator. In order for such ideal functionalities to be equivalent to the concrete protocol they are in most cases tailored to the particular protocol and it is difficult to provide several implementations for one, more abstract ideal functionality. The ideal functionalities given in [AG99] are comparable to the ideal functionality obtained when composed with a simulator in the setting of strong simulatability.*

6.3.3 Other notions of simulation based security

Several other notions of simulation based security appear in the literature. We present them, and show that they all coincide in our setting. This coincidence is regarded as highly desirable [KDMR08, Küs06], while it does not hold in most simulation-based security frameworks [Can01, BPW07].

Definition 6.8 *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two functionalities and $\mathcal{A}$ be any adversary for $\mathcal{F}_1$.*

- $\mathcal{F}_1$ *emulates* $\mathcal{F}_2$ *in the sense of* black box simulatability, $\mathcal{F}_1 \leq^{\mathsf{BB}} \mathcal{F}_2$, *iff* $\exists \mathcal{S}.\, \forall \mathcal{A}.\, \mathcal{A}[\mathcal{F}_1] \preceq \mathcal{A}[\mathcal{S}[\mathcal{F}_2]]$ *where* $\mathcal{S}$ *is an adversary for* $\mathcal{F}_2$ *with* $\mathsf{fnet}(S[\mathcal{F}_2]) = \mathsf{fnet}(\mathcal{F}_1)$.
- $\mathcal{F}_1$ *emulates* $\mathcal{F}_2$ *in the sense of* universally composable simulatability, $\mathcal{F}_1 \leq^{\mathsf{UC}} \mathcal{F}_2$, *iff* $\forall \mathcal{A}.\, \exists \mathcal{S}.\, \mathcal{A}[\mathcal{F}_1] \preceq \mathcal{S}[\mathcal{F}_2]$ *where* $\mathcal{S}$ *is an adversary for* $\mathcal{F}_2$ *s.t.* $\mathsf{fnet}(\mathcal{A}[\mathcal{F}_1]) = \mathsf{fnet}(\mathcal{S}[\mathcal{F}_2])$.
- $\mathcal{F}_1$ *emulates* $\mathcal{F}_2$ *in the sense of* universally composable simulatability with dummy adversary, $\mathcal{F}_1 \leq^{\mathsf{UCDA}} \mathcal{F}_2$, *iff* $\exists \mathcal{S}.\, D[\mathcal{F}_1] \preceq \mathcal{S}[\mathcal{F}_2]$ *where* D *is the dummy adversary for* $\mathcal{F}_1$ *and* $\mathcal{S}$ *is an adversary for* $\mathcal{F}_2$ *such that* $\mathsf{fnet}(\mathcal{S}[\mathcal{F}_2]) = \mathsf{fnet}(D[\mathcal{F}_1])$.

Theorem 6.3 *We have that* $\leq^{\mathsf{SS}} = \leq^{\mathsf{BB}} = \leq^{\mathsf{UC}} = \leq^{\mathsf{UCDA}}$.

The above security notions can also be defined replacing observational preorder by observational equivalence denoted $\leq^{\mathsf{SS}}_{\approx}, \leq^{\mathsf{BB}}_{\approx}, \leq^{\mathsf{UC}}_{\approx}$ and $\leq^{\mathsf{UCDA}}_{\approx}$. Surprisingly, the use of observational equivalence turns out to be too strong, ruling out natural secure emulation cases: for instance, the $\leq^{\mathsf{SS}}_{\approx}$ relation is not reflexive, due to the extra non-determinism that the simulator introduces. While symbolic analysis techniques typically rely on bisimulation relations, this is however consistent with the definitions proposed in the task-PIOA framework [CCK+07], which also allows non-deterministic executions for simulation based security.

6.3.4 Applications

In [24] we illustrate our framework by showing the secure emulation of a mutual authentication functionality by the Needham-Shroeder-Lowe (NSL) protocol [Low95]. As the NSL protocol uses public key encryption we first introduce real and ideal functionalities for asymmetric encryption.

The real public key encryption functionality $\mathcal{P}_{\mathsf{pke}}$ generates a new fresh private key sk when initialized and publishes the corresponding public key. Then the process is ready to receive encryption or decryption requests. The idealized version $\mathcal{F}_{\mathsf{pke}}$ differs from traditional ideal functionalities for asymmetric encryption which produce ciphertexts by encrypting random strings [BPW07, Can01, KT08]. An association table (plaintext/ciphertext) is then necessary to perform decryption. In our symbolic setting we avoid such a table (which would be cumbersome to encode) by using two layers of encryption and a secure key ssk. During the initialization of $\mathcal{F}_{\mathsf{pke}}$ the secret key sk is chosen by the adversary. However, neither $\mathsf{pk}(ssk)$ nor ssk are ever transmitted by $\mathcal{F}_{\mathsf{pke}}$, guaranteeing that it is impossible to decrypt such a ciphertext outside the functionality, even if the key sk is adversarially chosen. This will be a crucial feature for our *joint state composition* result which is the motivation to have an ideal encryption functionality in a symbolic setting. As expected, we show that $\mathcal{P}_{\mathsf{pke}} \leq^{\mathsf{SS}} \mathcal{F}_{\mathsf{pke}}$.

We will now explain our joint state result. While $\leq^{\mathsf{SS}}$ is stable under replication this is not always sufficient to obtain composition guarantees. Indeed replication of a process also replicates all key generation operations. In order to obtain self-composition and inter-protocol composition with common key material we need a *joint state functionality*, i.e. a functionality that realizes $!\mathcal{F}_{\mathsf{pke}}$ while reusing the same key material. We actually consider the functionality $\underline{\mathcal{F}_{\mathsf{pke}}}$, which is a variant of $\mathcal{F}_{\mathsf{pke}}$ in which each message is tagged. More precisely, the process $\underline{\mathcal{F}_{\mathsf{pke}}}$ is defined as $\mathcal{F}_{\mathsf{pke}}$, except that (i) the functionality additionally

inputs a session id *sid* during the initialization, (ii) each input and output contain this *sid* and the corresponding checks are performed.

We then define an IO context $\mathcal{P}_{\mathsf{js}}[\_]$ for the initial functionality $\mathcal{F}_{\mathsf{pke}}$. $\mathcal{P}_{\mathsf{js}}[\_]$ offers the same IO interface as the modified functionality $\underline{\mathcal{F}_{\mathsf{pke}}}$ and "translates" requests and responses by adding or removing the session identifier to the plaintexts that are submitted or received from $\mathcal{F}_{\mathsf{pke}}$. In that way the joint state functionality $\mathcal{P}_{\mathsf{js}}[\underline{\mathcal{F}_{\mathsf{pke}}}]$ uses a single instance of $\underline{\mathcal{F}_{\mathsf{pke}}}$ for all protocol sessions and tags the encryptions of each session using the session identifier, in a similar way as in the previous section. Our joint joint state result can be summarized as follows:

$$\begin{array}{ccc} \mathcal{P}_{\mathsf{js}}[\mathcal{F}_{\mathsf{pke}}] & \leq^{\mathsf{SS}} & !\underline{\mathcal{F}_{\mathsf{pke}}} \\ \mathsf{SS}\ \vee\mathsf{I} & & \mathsf{SS}\ \vee\mathsf{I} \\ \mathcal{P}_{\mathsf{js}}[\mathcal{P}_{\mathsf{pke}}] & \not\leq^{\mathsf{SS}} & !\underline{\mathcal{P}_{\mathsf{pke}}} \end{array}$$

Note in particular the importance of having defined an ideal encryption functionality where the attacker chooses the secret key *sk*. Indeed, $\mathcal{P}_{\mathsf{js}}[\mathcal{P}_{\mathsf{pke}}] \not\leq^{\mathsf{SS}} !\underline{\mathcal{P}_{\mathsf{pke}}}$ as the attacker can observe that different public keys are output in $!\underline{\mathcal{P}_{\mathsf{pke}}}$, while a unique public key is used in $\mathcal{P}_{\mathsf{js}}[\mathcal{P}_{\mathsf{pke}}]$. When using the ideal functionality, the "trick" is to have the simulator send the same *sk* to each copy of $\underline{\mathcal{F}_{\mathsf{pke}}}$.

In [24] we design a functionality for mutual authentication $\mathcal{F}_{\mathsf{auth}}$ which roughly works as follows. Both the initiator and the responder receive a request for mutual authentication. They forward this request to the adversary and, if both parties are honest, to a trusted host which compares these requests and authorizes going further if they match. Eventually, when the adversary asks to finish the protocol, then both participants complete the protocol session. We have shown that $\mathcal{F}_{\mathsf{auth}}$ can be realized by a functionality which implements the Needham-Schroeder-Lowe protocol [Low95]. Using the joint state result for asymmetric encryption we lift this emulation to hold for an unbounded number of sessions.

6.4 Conclusion and perspectives

In this chapter we have summarized our work on composition. Future work on composition can be structured in three themes.

Proving design principles sound. Many guidelines and principles suggesting good engineering practices for security protocols have been proposed, *e.g.*, [AN06, AN95, WL94, GS95]. Such principles provide valuable, yet informal guidelines and "thumb rules". Formalizing these guidelines and studying what guarantees they can ensure would be interesting. One such principle is that of *full information*, proposed by Woo and Lam [WL94], stating that any message should contain the history of all previous messages, using for instance hashes, such as in the SSL protocol. Another design methodology is that of *fail-stop* protocols which requires protocols to halt as soon as soon as there is any derivation from the designed execution path.

From trace to indistinguishability properties. Currently, most works on composition are limited to trace properties. It would be interesting to investigate whether these results can

be transferred to indistinguishability properties. Is it possible to find sufficient conditions such that

$$\textbf{if} \quad \nu k.P \sim \nu k.Q \quad \textbf{then} \quad \nu k.(P \mid R) \sim \nu k.(Q \mid R)$$

and similarly for self-composition

$$\textbf{if} \quad \nu k.P \sim Q \quad \textbf{then} \quad \nu k.!P \sim !Q$$

The proofs in [CD09b, 26] are based on a constraint solving procedure for deciding security properties. In recent work, Cheval *et al.* [CCD10] also use a constraint solving procedure to decide equivalence properties. A concrete question is whether the proofs of [CD09b, 26] can be adapted to this procedure to obtain composition results for indistinguishability properties.

Simulation based security. Our work on symbolic simulation based security is still in its beginning. We would like to explore the use of this setting on particular applications. A first application would be security APIs (which we will discuss in more detail in Chapter 4). Security properties expected from an API are often not clear and depend on the particular application which uses the API. Reasoning about an idealized API seems therefore a natural approach.

Another more ambitious application is electronic voting. As we have seen in Chapter 3, security properties of electronic voting are difficult to formalize. Again, an ideal functionality which does not need to explicit the properties seems appealing. However, in our current setting, as was pointed out by D. Unruh in a personal communication, anonymity properties may not be transferred from an ideal functionality to its implementation due to the use of an observational preorder instead of an observational equivalence. D. Unruh is currently working on an extension of our setting to overcome this limitation.

CHAPTER 7

Computational soundness

Since the 1980s, two approaches have been developed for analyzing security protocols. One of the approaches relies on a computational model that considers issues of complexity and probability. This approach captures a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. The other approach—which my previous work is part of—relies on a symbolic model of protocol executions in which cryptographic primitives are treated as "black boxes", in the sense that the attacker can only perform a given set of actions. Since the seminal work of Dolev and Yao, it has been realized that this latter approach enables significantly simpler and often automated proofs. Unfortunately, the high degree of abstraction and the limited adversary power raise questions regarding the security offered by such proofs. Potentially, justifying symbolic proofs with respect to standard computational models has tremendous benefits: protocols can be analyzed using automated tools and still benefit from the security guarantees of the computational model. In their seminal work, Abadi and Rogaway prove the computational soundness of formal (symmetric) encryption in the case a passive attacker. Since then, many results have been obtained. Each of these results considers a fixed set of primitives, for instance symmetric or public key encryption. In [35, 12], we design a general framework for comparing formal and computational models in the presence of a passive attacker. We define the notions of soundness and faithfulness of a cryptographic implementation with respect to equality, static equivalence and (non-)deducibility and present proof techniques for showing soundness results illustrated on several examples (xor, ciphers and lists). In [32] we generalise this framework to consider an adaptive adversary, which is strictly more powerful than a passive one. These result as well as a soundness result for a theory of bilinear pairings [11] will be presented in the following sections.

Computational and symbolic models also differ on the way security properties are specified. The standard symbolic, deducibility-based notions of secrecy are in general insufficient from a cryptographic point of view, especially in presence of hash functions. In [33] we consider an active adversary and devise a more appropriate secrecy criterion which exactly captures a standard cryptographic notion of secrecy for protocols involving public-key encryption and hash functions: protocols that satisfy it are computationally

secure while any violation of our criterion directly leads to an attack. We briefly summarize this work in Section 7.3.

During the last years there has been a large amount of work on this topic. In this chapter we will only mention a small fragment of the papers on symbolic methods for computational proofs of cryptographic protocols. For a more intensive survey we refer the reader to [7].

7.1 Computational algebras

We now give terms and frames a concrete semantics, parametrized by an implementation of the primitives. Provided a set of sorts $\mathcal{S}$ and a set of function symbols $\mathcal{F}$, a *$(\mathcal{S},\mathcal{F})$-computational algebra* A consists of

- a non-empty set of bit-strings $[\![s]\!]_A \subseteq \{0,1\}^*$ for each sort $s \in \mathcal{S}$;
- a computable function $[\![f]\!]_A : [\![s_1]\!]_A \times \ldots \times [\![s_k]\!]_A \to [\![s]\!]_A$ for each $f \in \mathcal{F}$ with $\mathit{arity}(f) = s_1 \times \ldots \times s_k \to s$;
- an effective procedure to draw random elements from $[\![s]\!]_A$; we denote such a drawing by $x \xleftarrow{R} [\![s]\!]_A$;

Assume a fixed $(\mathcal{S},\mathcal{F})$-computational algebra A. We associate to each frame $\varphi = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ a distribution $\psi = [\![\varphi]\!]_A$, of which the drawings $\widehat{\psi} \xleftarrow{R} \psi$ are computed as follows:

1. for each name a of sort s appearing in $t_1, \ldots, t_n$, draw a value $\widehat{a} \xleftarrow{R} [\![s]\!]_A$;
2. for each x_i $(1 \le i \le n)$ of sort s_i, compute $\widehat{t_i} \in [\![s_i]\!]_A$ recursively on the structure of terms: $\widehat{f(t'_1, \ldots, t'_m)} = [\![f]\!]_A(\widehat{t'_1}, \ldots, \widehat{t'_m})$;
3. return the value $\widehat{\psi} = \{x_1 \mapsto \widehat{t_1}, \ldots, x_n \mapsto \widehat{t_n}\}$.

Such values $\phi = \{x_1 = e_1, \ldots, x_n = e_n\}$ with $e_i \in [\![s_i]\!]_A$ are called *concrete frames*. We extend the notation $[\![.]\!]_A$ to (tuples of) closed terms in the obvious way.

In the following we focus on asymptotic notions of cryptographic security and consider families of computational algebra (A_η) indexed by a complexity parameter $\eta \ge 0$. (This parameter η might be thought of as the size of keys and other secret values.) The *concrete semantics* of a frame φ is a family of distributions over concrete frames $([\![\varphi]\!]_{A_\eta})$. We only consider families of computational algebras (A_η) such that each required operation on algebras is feasible by a (uniform, probabilistic) polynomial-time algorithm in the complexity parameter η. This ensures that the concrete semantics of terms and frames is efficiently computable (in the same sense).

Families of distributions (*ensembles*) over concrete frames benefit from the usual notion of cryptographic indistinguishability. Intuitively, two families of distributions (ψ_η) and (ψ'_η) are *indistinguishable*, written $(\psi_\eta) \approx (\psi'_\eta)$, iff no probabilistic polynomial-time adversary

$\mathcal{A}$ can guess whether he is given a sample from ψ_η or ψ'_η with a probability significantly greater than $\frac{1}{2}$. Formally, we ask the *advantage* of $\mathcal{A}$,

$$\mathcal{A}^{\mathrm{IND}}_{\mathcal{A}}(\psi_\eta, \psi'_\eta) = \left|\mathbb{P}[\widehat{\psi} \xleftarrow{R} \psi_\eta \ : \ \mathcal{A}(\widehat{\psi}) = 1] - \mathbb{P}[\widehat{\psi} \xleftarrow{R} \psi'_\eta \ : \ \mathcal{A}(\widehat{\psi}) = 1]\right|$$

to be a *negligible* function of η, that is, to remain eventually smaller than any η^{-n} $(n > 0)$ for sufficiently large η.

7.2 Soundness of equational theories

7.2.1 Definitions of soundness and faithfullness

We introduce the notions of sound and faithful computational algebras with respect to the formal relations studied here: equality, static equivalence and deducibility.

Definition 7.1 *Let $\mathcal{E}$ be an equational theory. A family of computational algebras (A_η) is*

- $=_\mathcal{E}$-sound *iff for every closed terms T_1, T_2 of the same sort, $T_1 =_\mathcal{E} T_2$ implies that $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1, T_2]\!]_{A_\eta} \ : \ e_1 =_{A_\eta} e_2]$ is overwhelming;*
- $=_\mathcal{E}$-faithful *iff for every closed terms T_1, T_2 of the same sort, $T_1 \neq_\mathcal{E} T_2$ implies that $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1, T_2]\!]_{A_\eta} \ : \ e_1 =_{A_\eta} e_2]$ is negligible;*
- $\sim_\mathcal{E}$-sound *iff for every frames φ_1, φ_2 with the same domain, $\varphi_1 \sim_\mathcal{E} \varphi_2$ implies that $([\![\varphi_1]\!]_{A_\eta}) \approx ([\![\varphi_2]\!]_{A_\eta})$;*
- $\sim_\mathcal{E}$-faithful *iff for every frames φ_1, φ_2 of the same domain, $\varphi_1 \not\sim_\mathcal{E} \varphi_2$ implies that there exists a polynomial-time adversary $\mathcal{A}$ for distinguishing concrete frames, such that $\mathcal{A}^{IND}(\mathcal{A}, \eta, [\![\varphi_1]\!]_{A_\eta}, [\![\varphi_2]\!]_{A_\eta})$ is overwhelming;*
- $\nvdash_\mathcal{E}$-sound *iff for every frame φ and closed term T such that $names(T) \subseteq names(\varphi)$, $\varphi \nvdash_\mathcal{E} T$ implies that for each polynomial-time adversary $\mathcal{A}$, $\mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta} : \mathcal{A}(\phi) =_{A_\eta} e]$ is negligible;*
- $\nvdash_\mathcal{E}$-faithful *iff for every frame φ and closed term T such that $names(T) \subseteq names(\varphi)$, $\varphi \vdash_\mathcal{E} T$ implies that there exists a polynomial-time adversary $\mathcal{A}$ such that $\mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta} \ : \ \mathcal{A}(\phi) =_{A_\eta} e]$ is overwhelming.*

Sometimes, it is possible to prove stronger notions of soundness that hold without restriction on the computational power of adversaries. We call these notions *unconditional*.

Definition 7.2 *Let $\mathcal{E}$ be an equational theory. A family of computational algebras (A_η) is*

- unconditionally $=_\mathcal{E}$-sound *iff for every closed terms T_1, T_2 of the same sort, $T_1 =_\mathcal{E} T_2$ implies that $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1, T_2]\!]_{A_\eta} \ : \ e_1 =_{A_\eta} e_2] = 1$;*
- unconditionally $\sim_\mathcal{E}$-sound *iff for every frames φ_1, φ_2 with the same domain, $\varphi_1 \sim_\mathcal{E} \varphi_2$ implies $([\![\varphi_1]\!]_{A_\eta}) = ([\![\varphi_2]\!]_{A_\eta})$;*

- unconditionally $\not\vdash_{\mathcal{E}}$-sound *iff for every frame* φ *and closed term* T *such that* $names(T) \subseteq names(\varphi)$ *and* $\varphi \not\vdash_{\mathcal{E}} T$*, the drawings for* φ *and* T *are independent: for all* ϕ_0*,* e_0*, we have that* $\mathbb{P}[\phi_0, e_0 \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta}] = \mathbb{P}[\phi_0 \xleftarrow{R} [\![\varphi]\!]_{A_\eta}] \times \mathbb{P}[e_0 \xleftarrow{R} [\![T]\!]_{A_\eta}]$*, and the drawing* $(\xleftarrow{R} [\![T]\!]_{A_\eta})$ *is collision-free.*

The fact that the first two unconditional notions are stronger than their computational counterparts is clear from the definitions. As for the unconditional $\not\vdash_{\mathcal{E}}$-soundness, observe that if the drawings for φ and T are independent, and the drawing $(\xleftarrow{R} [\![T]\!]_{A_\eta})$ is collision-free, then any adversary $\mathcal{A}$ has negligible probability of retrieving the value of T:

$$\begin{aligned}\mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta} : \mathcal{A}(\phi) =_{A_\eta} e] &= \mathbb{P}[\phi \xleftarrow{R} [\![\varphi]\!]_{A_\eta}, e \xleftarrow{R} [\![T]\!]_{A_\eta} : \mathcal{A}(\phi) =_{A_\eta} e] \\ &\leq \sup_{e_0} \mathbb{P}[e \xleftarrow{R} [\![T]\!]_{A_\eta} : e =_{A_\eta} e_0]\end{aligned}$$

Generally, (unconditional) $=_{\mathcal{E}}$-soundness is given by construction. Indeed true formal equations correspond to the expected behavior of primitives and should hold in the concrete world with overwhelming probability. The other criteria are however more difficult to fulfill. Therefore it is often interesting to restrict frames to *well-formed* ones in order to achieve soundness or faithfulness: for instance Abadi and Rogaway [AR02] do forbid encryption cycles.

It is worth noting that the notions of soundness and faithfulness introduced above are not independent.

Proposition 7.1 *Let* (A_η) *be a* $=_{\mathcal{E}}$*-sound family of computational algebras. Then*

1. (A_η) *is* $\not\vdash_{\mathcal{E}}$*-faithful;*
2. *if* (A_η) *is also* $=_{\mathcal{E}}$*-faithful,* (A_η) *is* $\sim_{\mathcal{E}}$*-faithful.*

Proposition 7.2 *Let* (A_η) *be a family of* $\sim_{\mathcal{E}}$*-sound computational algebras. Assume that free binary symbols* $\mathsf{h}_s : s \times Key \to Hash$ *are available for every sort* s*, where the sort* Key *is not degenerated in* $\mathcal{E}$*, and the drawing of random elements for the sort* $Hash$*,* $(\xleftarrow{R} [\![Hash]\!]_{A_\eta})$*, is collision-free. Then*

1. (A_η) *is* $=_{\mathcal{E}}$*-faithful;*
2. (A_η) *is* $\not\vdash_{\mathcal{E}}$*-sound;*
3. *Assume the implementations for the symbols* h_s *are* collision-resistant*, that is, assume that for all* T_1, T_2 *of sort* s*, given a fresh name* k *of sort* Key*, the quantity*

$$\mathbb{P}\left[e_1, e_2, e_1', e_2' \xleftarrow{R} [\![T_1, T_2, \mathsf{h}_s(T_1, k), \mathsf{h}_s(T_2, k)]\!]_{A_\eta} \ : \ e_1 \neq_{A_\eta} e_2,\ e_1' =_{A_\eta} e_2'\right]$$

is negligible. Then (A_η) *is* $=_{\mathcal{E}}$*-sound,* $\not\vdash_{\mathcal{E}}$*-faithful and* $\sim_{\mathcal{E}}$*-faithful.*

We extend soundness of static equivalence to the adaptive setting from [MP05]. In $\sim_{\mathcal{E}}$-soundness the adversary observes the computational value of a fixed frame whereas in

this setting the adversary sees the computational value of a sequence of adaptively chosen frames.

The adaptive setting is formalized through the following cryptographic game. Let (A_η) be a family of computational algebras and $\mathcal{A}$ be an adversary. $\mathcal{A}$ has access to a left-right evaluation oracle $\mathcal{O}_{LR}$ which given a pair of symbolic terms (t_0, t_1) outputs either the implementation of t_0 or of t_1. This oracle depends on a selection bit b and uses a local store in order to record values generated for the different names (these values are used when processing further queries). With a slight abuse of notation, we omit this store and write:

$$\mathcal{O}^b_{LR,A_\eta}(t_0, t_1) = [\![t_b]\!]_{A_\eta}$$

Adversary $\mathcal{A}$ plays an indistinguishability game and its objective is to find the value of b. Formally the advantage of $\mathcal{A}$ is defined by:

$$\mathcal{A}^{\mathsf{ADPT}}_{\mathcal{A},A_\eta}(\eta) = \left|\mathbb{P}[\mathcal{A}^{\mathcal{O}^1_{LR,A_\eta}} = 1] - \mathbb{P}[\mathcal{A}^{\mathcal{O}^0_{LR,A_\eta}} = 1]\right|$$

Without further restrictions on the queries made by the adversary, having a non-negligible advantage is easy in most cases. For example the adversary could submit a pair $(0, 1)$ to his oracle. We therefore require the adversary to be *legal*.

Definition 7.3 (Adaptive soundness) *An adversary $\mathcal{A}$ is* legal *if for any sequence of queries $(t^i_0, t^i_1)_{1\leq i\leq n}$ made by $\mathcal{A}$ to its left-right oracle, queries are statically equivalent:*

$$\left\{x_1 \mapsto t^1_0, \ldots, x_n \mapsto t^n_0\right\} \sim_{\mathcal{E}} \left\{x_1 \mapsto t^1_1, \ldots, x_n \mapsto t^n_1\right\}$$

A family of computational algebras (A_η) is

- $\sim_{\mathcal{E}}$-ad-sound *iff the advantage $\mathcal{A}^{\mathsf{ADPT}}_{\mathcal{A},A_\eta}(\eta)$ of any polynomial-time legal adversary $\mathcal{A}$ is negligible.*
- unconditionally $\sim_{\mathcal{E}}$-ad-sound *iff the advantage $\mathcal{A}^{\mathsf{ADPT}}_{\mathcal{A},A_\eta}(\eta)$ of any legal adversary $\mathcal{A}$ is* 0.

Note that as variables are typed, any query (t^i_0, t^i_1) of a legal adversary to the oracle is such that t^i_0 and t^i_1 have the same sort. Adaptive soundness implies the original soundness notion for static equivalence.

Proposition 7.3 *Let (A_η) be a family of computational algebras. If A_η is $\sim_{\mathcal{E}}$-ad-sound then A_η is also $\sim_{\mathcal{E}}$-sound but the converse is false in general.*

Proposition 7.4 *Let (A_η) be a family of computational algebras. A_η is unconditionally $\sim_{\mathcal{E}}$-ad-sound iff A_η is unconditionally $\sim_{\mathcal{E}}$-sound.*

7.2.2 Soundness of several theories

Symmetric encryption We consider the case of symmetric encryption as in the models from [AR02] and from [MP05]. Hence we assume that the implementation of the symmetric encryption scheme is semantically secure [GM82].

Our symbolic model consists of the set of sorts $\mathcal{S} = \{Data\}$, an infinite number of names for sort $Data$ called keys and the function symbols:

$\mathsf{senc}, \mathsf{sdec}$	:	$Data \times Data \to Data$	encrypt, decrypt
pair	:	$Data \times Data \to Data$	pair constructor
$\mathsf{proj}_1, \mathsf{proj}_2$	:	$Data \to Data$	projections
$\mathsf{samekey}$	:	$Data \times Data \to Data$	key equalities test
$\mathsf{tenc}, \mathsf{tpair}$	:	$Data \to Data$	type testers
$0, 1$	:	$Data$	constants

A name k is used at a key position in a term t if there exists a sub-term $\mathsf{senc}(t', k)$ of t. Else k is used at a plaintext position. We consider the equational theory $\mathcal{E}_{\mathsf{sym}}$ generated by:

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & = & x & \mathsf{samekey}(\mathsf{senc}(x,y),\mathsf{senc}(z,y)) & = & 1 \\ \mathsf{proj}_1(\mathsf{pair}(x,y)) & = & x & \mathsf{tenc}(\mathsf{senc}(x,y)) & = & 1 \\ \mathsf{proj}_2(\mathsf{pair}(x,y)) & = & y & \mathsf{tpair}(\mathsf{pair}(x,y)) & = & 1 \end{array}$$

The importance of key cycles was already described in [AR02]. In general INDCPA is not sufficient to prove any soundness result in the presence of key cycles. Thus, as in numerous previous work, we forbid the formal terms to contain such cycles. Let $\prec$ be a total order among keys. A *frame φ is acyclic for $\prec$* if for any subterm $\mathsf{senc}(t, k)$ of φ, if k' occurs in t then $k' \prec k$. Moreover as noted in [MP05], selective decommitment [DNRS03] can be a problem. The classical solution to this problem is to require keys to be sent *before* being used to encrypt a message or they must never appear as a plaintext. A *frame* $\varphi = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ *is well-formed for* $\prec$ if

- φ is acyclic for $\prec$;
- the terms t_i only use symbols senc, pair, 0 and 1, and only names are used at key positions;
- if k is used as plaintext in t_i, then k cannot be used at a key position in t_j for $j < i$.

An *adversary is well-formed for* $\prec$ if the sequence of queries $(t_0^i, t_1^i)_{1 \le i \le n}$ that he makes to his oracle yields two well-formed frames $\{x_1 \mapsto t_0^1, \ldots, x_n \mapsto t_0^n\}$ and $\{x_1 \mapsto t_1^1, \ldots, x_n \mapsto t_1^n\}$ for $\prec$.

We suppose that senc is implemented by a symmetric encryption scheme $\mathcal{SE}$ and that pairing is implemented by some usual implementation that realizes a pairing operation. We suppose that INDCPA is a length-concealing variant of semantic security and INDCPA′ is a non-adaptive version of INDCPA, i.e., the adversary may make one single query, but submitting two lists of plaintexts, rather that multiple queries consisting each of a pair of plaintexts.

Theorem 7.1 *Let $\prec$ be a total order among keys. In the remainder of this proposition we only consider well-formed adversaries for $\prec$. Let (A_η) be a family of computational algebras based on a symmetric encryption scheme $\mathcal{SE}$.*

- *(A_η) is $\sim_{\mathcal{E}_{\mathsf{sym}}}$-ad-sound if $\mathcal{SE}$ is* INDCPA *but the converse is false.*
- *(A_η) is $\sim_{\mathcal{E}_{\mathsf{sym}}}$-sound if $\mathcal{SE}$ is* INDCPA′ *but the converse is false.*

Ciphers and lists We now detail the example of symmetric, deterministic and length-preserving encryption schemes. Such schemes, also known as *pseudo-random permutations* or *ciphers* [PP04], are widely used in practice, the most famous examples (for fixed-length inputs) being DES and AES.

Our formal model consists of a set of sorts $\mathcal{S} = \{Data, Data_0, Data_1 \dots Data_n \dots\}$, an infinite number of names for every sort $Data$ and $Data_n$, and the following symbols (for every $n \geq 0$):

$\mathsf{enc}_n, \mathsf{dec}_n$	:	$Data_n \times Data \to Data_n$	encryption, decryption
cons_n	:	$Data \times Data_n \to Data_{n+1}$	list constructor
head_n	:	$Data_{n+1} \to Data$	head of a list
tail_n	:	$Data_{n+1} \to Data_n$	tail of a list
nil	:	$Data_0$	empty list
$0, 1$	:	$Data$	constants

We consider the equational theory $\mathcal{E}_{\mathsf{cipher}}$ generated by the following equations (for every $n \geq 0$ and for every name a_0 of sort $Data_0$):

$$\begin{array}{rcl} \mathsf{dec}_n(\mathsf{enc}_n(x,y),y) & = & x \\ \mathsf{enc}_n(\mathsf{dec}_n(x,y),y) & = & x \\ \mathsf{head}_n(\mathsf{cons}_n(x,y)) & = & x \\ \mathsf{tail}_n(\mathsf{cons}_n(x,y)) & = & y \\ \mathsf{cons}_n(\mathsf{head}_n(x),\mathsf{tail}_n(x)) & = & x \end{array} \qquad \begin{array}{rcl} \mathsf{enc}_0(\mathsf{nil},x) & = & \mathsf{nil} \\ \mathsf{dec}_0(\mathsf{nil},x) & = & \mathsf{nil} \\ \mathsf{tail}_0(x) & = & \mathsf{nil} \\ a_0 & = & \mathsf{nil} \end{array}$$

where x, y are variables of the appropriate sorts in each case. The effect of the last four equations is that the sort $Data_0$ is degenerated in $\mathcal{E}_{\mathsf{cipher}}$, that is, all terms of sort $Data_0$ are equal. Notice that each term has a unique sort. As the subscripts n of function symbols are redundant with sorts, we tend to omit them in terms. For instance, if $k, k' : Data$, we may write $\mathsf{enc}(\mathsf{cons}(k, \mathsf{nil}), k')$ instead of $\mathsf{enc}_1(\mathsf{cons}_0(k, \mathsf{nil}), k')$.

The concrete meaning of sorts and symbols is given by the computational algebras A_η, $\eta > 0$, defined as follows:

- the carrier sets are $[\![Data]\!]_{A_\eta} = \{0,1\}^\eta$ and $[\![Data_n]\!]_{A_\eta} = \{0,1\}^{n\eta}$ equipped with the uniform distribution and the usual equality relation;
- $\mathsf{enc}_n, \mathsf{dec}_n$ are implemented by a cipher for data of size $n\eta$ and keys of size η;
- $[\![\mathsf{nil}]\!]_{A_\eta}$ is the empty bit-string, $[\![\mathsf{cons}_n]\!]_{A_\eta}$ is the usual concatenation, $[\![0]\!]_{A_\eta} = 0^\eta$, $[\![1]\!]_{A_\eta} = 1^\eta$, $[\![\mathsf{head}_n]\!]_{A_\eta}$ returns the η first digits of bit-strings (of size $(n+1)\eta$) whereas $[\![\mathsf{tail}_n]\!]_{A_\eta}$ returns the last $n\eta$ digits.

As previously, we restrict frames to those with only atomic keys and no encryption cycles. A closed frame is well-formed iff it has only atomic keys, contains no encryption cycles and uses no head and tail symbols. We have shown the following soundness result given a classical cryptographic assumptions for ciphers, super pseudo-random permutation (SPRP) [PP04].

Theorem 7.2 ($\sim_{\mathcal{E}_{\mathsf{cipher}}}$-soundness) *Let φ_1 and φ_2 be two well-formed frames of the same domain. Assume that the concrete implementations for the encryption satisfies the SPRP assumption. If $\varphi_1 \sim_{\mathcal{E}_{\mathsf{cipher}}} \varphi_2$ then $([\![\varphi_1]\!]_{A_\eta}) \approx ([\![\varphi_2]\!]_{A_\eta})$.*

Exclusive Or We study the soundness and faithfulness problems for the natural theory and implementation of the exclusive OR (XOR), together with constants and (pure) random numbers.

The formal model consists of a single sort *Data*, an infinite number of names, the infix symbol $\oplus : Data \times Data \to Data$ and two constants $0, 1 : Data$. Terms are equipped with the equational theory $\mathcal{E}_\oplus$ generated by:

$$\begin{array}{rclcrcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) & & x \oplus x & = & 0 \\ x \oplus y & = & y \oplus x & & x \oplus 0 & = & x \end{array}$$

As an implementation, we define the computational algebras A_η, $\eta \geq 0$:

- the concrete domain $[\![Data]\!]_{A_\eta}$ is the set of bit-strings of length η, $\{0,1\}^\eta$, equipped with the uniform distribution;
- $\oplus$ is interpreted by the usual XOR function over $\{0,1\}^\eta$;
- $[\![0]\!]_{A_\eta} = 0^\eta$ and $[\![1]\!]_{A_\eta} = 1^\eta$.

Theorem 7.3 *The usual implementation of XOR for the considered signature, (A_η), is unconditionally $=_{\mathcal{E}_\oplus}$-, $\sim_{\mathcal{E}_\oplus}$- and $\not\vdash_{\mathcal{E}_\oplus}$-sound. It is also $=_{\mathcal{E}_\oplus}$-, $\sim_{\mathcal{E}_\oplus}$- and $\not\vdash_{\mathcal{E}_\oplus}$-faithful.*

From Proposition 7.4 unconditional $\sim_{\mathcal{E}}$-ad-sound directly follows as one would expect for this theory.

In [32] we have also shown adaptive soundness of static equivalence for a joint, hierarchical theory of encryption and exclusive or where the exclusive or cannot appear on top of an encryption.

Diffie-Hellman exponentiation As a further application, we study soundness of modular exponentiation. The cryptographic assumption we make is that the *Decisional Diffie-Hellman* (DDH) problem is difficult: even when given g^x and g^y, it is difficult for any feasible computation to distinguish between g^{xy} and g^r, when x, y and r are selected at random. The original Diffie-Hellman protocol [DH76] has been used as a building block for several key agreement protocols that are widely used in practice (e.g. SSL/TLS and Kerberos V5) as well as for group key exchange protocols such as AKE1 [BCP01] or the Burmester-Desmedt protocol [BD94].

The symbolic model consists of two sorts G (for group elements) and R (for ring elements), an infinite number of names for R (but no name for sort G) and the symbols:

$$\begin{array}{rcll} \exp & : & R \to G & \text{exponentiation} \\ +, \cdot & : & R \times R \to R & \text{add, mult} \\ - & : & R \to R & \text{inverse} \\ * & : & G \times G \to G & \text{mult in } \mathbb{G} \\ 0_R, 1_R & : & R & \text{constants} \end{array}$$

We consider the equational theory $\mathcal{E}_{\mathsf{dh}}$ generated by:

$$\begin{array}{ccc} x + y = y + x & x \cdot y = y \cdot x & (x + y) + z = x + (y + z) \\ x \cdot (y + z) = x \cdot y + x \cdot z & (x \cdot y) \cdot z = x \cdot (y \cdot z) & x + (-x) = 0_R \\ 0_R + x = x & 1_R \cdot x = x & \exp(x) * \exp(y) = \exp(x + y) \end{array}$$

We put two restrictions on formal terms: products have to be *power-free*, *i.e.*, x^n is forbidden for $n > 1$, and products must not contain more than l elements for some fixed bound l, i.e. $x_1 \cdot ... \cdot x_n$ is forbidden for $n > l$. Both restrictions come from the DDH assumption and seem difficult to avoid [BLMW07]. Furthermore we are only interested in frames using terms of sort G.

We show that the DDH assumption is necessary and sufficient to prove adaptive soundness.

Theorem 7.4 *Let (A_η) be a family of computational algebras. (A_η) is $\sim_{\mathcal{E}_{\mathsf{dh}}}$-sound iff (A_η) satisfies the* DDH *assumption. (A_η) is $\sim_{\mathcal{E}_{\mathsf{dh}}}$-ad-sound iff (A_η) satisfies the* DDH *assumption.*

The proof mainly relies on the 3DH assumption [BLMW07] which has been shown equivalent to DDH.

As for exclusive or, in [32] we have shown adaptive soundness of static equivalence for a joint, hierarchical theory of encryption and modular exponentiation where the exponentiations can either appear in key or plaintext positions.

Bilinear pairing In a slightly different setting which we will not introduce here we have also shown a soundness result for a model of *bilinear pairing* and symmetric encryption [11]. The result was shown for a *pattern equivalence* ($\cong$), as in [AR02], rather than static equivalence. The result holds holds under the bilinear DDH assumption and relies on similar proof techniques as for modular exponentiation [BLMW07].

Theorem 7.5 ([11]) *Let t_0 and t_1 be two acyclic well-formed terms, such that $t_0 \cong t_1$. Let $\mathcal{SE}$ be a symmetric encryption scheme that is secure for* INDCPA *and IG be an instance generator satisfying* BDDH*, then $[\![t_0]\!]_{\mathcal{SE},IG} \approx [\![t_1]\!]_{\mathcal{SE},IG}$.*

We have applied our soundness result to prove computational security of Joux's tripartite Diffie-Hellman protocol [Jou00], and Al-Riyami and Paterson's TAK-2 and TAK-3 protocols [ARP03]. By computational security we mean that the generated key is indistinguishable from a random element. To illustrate the scope of our result we also designed a new pairing based variant of the Burmester-Desmedt [BD94] protocol and prove its security in the passive setting.

7.3 Computational sound symbolic secrecy

In this section we briefly summarize our result on computationally sound secrecy for asymmetric encryption and hash functions in the presence of an active adversary. These results were obtained in a role-based model close to the one defined in [CW05]. We will not go into the details of this model which are not important for the presentation of the results.

For some security notions like integrity and authentication, the derivation of computational guarantees out of symbolic ones can be done with relative simplicity [BJ03, MW04]. In contrast, analogous results for the basic notion of secrecy proved significantly more elusive and have appeared only recently [BP05b, CW05, GS05, CH06]. The apparent reason for this situation is the striking difference between the definitional ideas used in the two different models. Symbolic secrecy typically states that the adversary cannot deduce the

entire secret from the messages it gathers in an execution. On the other hand, computational secrecy requires that not only the secret, but also no partial information is leaked to the adversary. A typical formulation that is used requires the adversary to distinguish between the secret and a completely unrelated alternative.

In [33] we investigate soundness results for symbolic secrecy in the presence of hash functions. One of the main motivations for considering hash functions, which have not been considered in the aforementioned results, is that they present a new challenge in linking symbolic and cryptographic secrecy: unlike ciphertexts, hashes have to be publicly verifiable, *i.e.*, any third party can verify if a value h is the hash value corresponding to a given message m. This implies that a simple-minded extension of previous results on symbolic and computational secrecy fails. Assume, for example, that in some protocol the hash $h = \mathsf{h}(s)$ of some secret s is sent in clear over the network. Then, while virtually all symbolic models would conclude that s remains secret (and this is also a naive assumption often made in practice), a trivial attack works in computational models: given s, s' and h, compare h with $\mathsf{h}(s)$ and $\mathsf{h}(s')$, and therefore recover s. Similar verifiability properties also occur in other settings, e.g. digital signatures which do not reveal the message signed.

We propose a new symbolic definition for nonce secrecy in protocols that use party identities, nonces, hash functions, and public key encryption. The definition that we give is based on the intuitively appealing concept of patterns [AR02].

Definition 7.4 (Patterns) *Given a set of closed terms $\phi = \{M_1, M_2, \ldots, M_k\}$ and a term T, we define $\mathsf{Pat}_T(\phi) = \{\mathsf{Pat}^\phi_T(M_1), \mathsf{Pat}^\phi_T(M_2), \ldots, \mathsf{Pat}^\phi_T(M_k)\}$, where $\mathsf{Pat}^\phi_T(M)$ defined recursively by:*

$$
\begin{array}{rcl}
\mathsf{Pat}^\phi_T(a) & = & \begin{cases} a & \text{if } \phi, T \vdash a \\ \Box & \text{otherwise} \end{cases} \\
\mathsf{Pat}^\phi_T(\langle M_1, M_2 \rangle) & = & \langle \mathsf{Pat}^\phi_T(M_1), \mathsf{Pat}^\phi_T(M_2) \rangle \\
\mathsf{Pat}^\phi_T(\{M\}^r_{\mathsf{pk}(a)}) & = & \begin{cases} \{\mathsf{Pat}^\phi_T(M)\}^r_{\mathsf{pk}(a)} & \text{if } \phi, T \vdash sk(a) \text{ or if } r \in \mathsf{Rand}_{adv} \\ \Box & \text{otherwise} \end{cases} \\
\mathsf{Pat}^\phi_T(h(M)) & = & \begin{cases} h(\mathsf{Pat}^\phi_T(M)) & \text{if } \phi, T \vdash M \\ \Box & \text{otherwise} \end{cases}
\end{array}
$$

Pat^ϕ_T is extended to set of messages as expected: $\mathsf{Pat}^\phi_T(S) = \bigcup_{t \in S} \mathsf{Pat}^\phi_T(t)$.

Given a protocol Π we denote by $\mathsf{Msg}^s(\Pi)$ the set of sets of messages which can be learned by the adversary in a valid symbolic executions. We can then define nonce secrecy based on this notion of patterns.

Definition 7.5 (Nonce secrecy) *Let Π be a protocol and $X^j_{A_i}$ a nonce variable occurring in some role A_i. We say that $X^j_{A_i}$ is secret in Π and we write $\Pi \models^s \mathsf{SecNonce}(i, j)$, if for every valid set of messages $\phi \in \mathsf{Msg}^s(\Pi)$ it holds that for every session number s, the symbolic nonce $n^{a_i,j,s}$ does not occur in $\mathsf{Pat}_{n^{a_i,j,s}}(\phi)$.*

The central aspect of our criterion is that it captures precisely security in the computational world in the sense that it is both sound and complete. More specifically, nonces

that are secret according to our *symbolic* criterion are also secret according to a standard *computational* definition. Furthermore, there exist successful attacks against the secrecy of any nonce that does not satisfy our definition. The computational definition of nonce secrecy is defined through the following experiment.

Definition 7.6 *Consider the experiment* $\mathbf{Exp}^{\mathrm{sec\_}b}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta)$ *parametrized by a bit* b *and that involves an adversary* $\mathcal{A}$ *against protocol* Π*. The experiment takes as input a security parameter* η *and starts by generating two random nonces* n_0 *and* n_1*. Then the adversary* $\mathcal{A}$ *starts interacting with the protocol* Π*. At some point in the execution the adversary initiates a session* s *in which the role of* A_i *is executed, and declares this session under attack. In this session, the variable* $X^j_{A_i}$ *is instantiated with* n_b*. At some point the adversary requires the two nonces* n_0 *and* n_1 *and has to output a guess* d*. The bit* d *is the result of the experiment. We define the advantage of the adversary* $\mathcal{A}$ *by:*

$$\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta) = \mathbb{P}[\mathbf{Exp}^{\mathrm{sec\_}1}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1] - \mathbb{P}[\mathbf{Exp}^{\mathrm{sec\_}0}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1]$$

We say that nonce $X^j_{A_i}$ *is computationally secret in protocol* Π*, and we write* $\Pi \models^c \mathsf{SecNonce}(i,j)$ *if for every p.p.t. adversary* $\mathcal{A}$ *its advantage is negligible.*

Theorem 7.6 *Let* Π *be an executable protocol and let* $X^j_{A_i}$ *be a nonce variable occurring in some role* A_i*. If the encryption scheme* $\mathcal{AE}$ *used in the implementation of* Π *is* INDCCA *secure and the hash function is interpreted by a random oracle then* $\Pi \models^s \mathsf{SecNonce}(i,j)$ *if and only if* $\Pi \models^c \mathsf{SecNonce}(i,j)$*.*

In the proofs we combine different techniques from cryptography and make direct use of a (non-trivial) extension of the mapping theorem of [MW04] to hash functions.

Our second important result is to prove the decidability of our symbolic secrecy criterion (w.r.t. a bounded number of sessions). This is a crucial result that enables the automatic verification of computational secrecy for nonces. We give an NP-decision procedure based on constraint solving, a technique that is suitable for practical implementations [ABB+05]. While the constraint solving technique is standard in automatic protocol analysis, we had to adapt it for our symbolic secrecy criterion: for the standard deducibility-based secrecy definition it suffices to transform constraint systems until one obtains a so-called simple form (on which satisfiable of the constraints is trivially decidable). However, for our symbolic secrecy criterion further transformations might be required in order for the procedure to be complete. Identifying a sufficient set of such transformations and proving that they are sufficient turned out to be non-trivial.

Related Work. We shortly compare this results to some related work. The papers that are immediately related to our work are those of Cortier and Warinschi [CW05], Backes and Pfitzmann [BP05b], and Canetti and Herzog [CH06], who study computationally sound secrecy properties, as well as the paper by Janvier *et al.* [JLM06], that presents a soundness result in the presence of hash functions. In this context, our work is the first to tackle computationally sound secrecy in the presence of hashes. We study the translation of symbolic secrecy into a computational version in a setting closely related to that in [CW05]. However, the use of hashes requires new notions and non-trivial extensions of the results

proved there. In [JLM06], Janvier *et al.* present a soundness result that differs however from this one. On the one hand they do not consider computational secrecy of nonces sent under hash functions. On the other hand, they present a new security criterion for hash functions, which is not the random oracle, although no implementation of a hash function satisfying their criterion is currently known. The work in [BP05b] and [CH06] is concerned with secrecy properties of key-exchange protocols in the context of simulation-based security, and hence, they study different computational settings. Interestingly, the symbolic criterion used in [CH06] is also formalized using patterns, but their use is unrelated to ours. None of the mentioned works considers decidability issues. In a more recent paper [CC08], Comon-Lundh and Cortier present a soundness result for observational equivalence in (a fragment of) the applied calculus for an equational theory of symmetric encryption. Observational equivalence obviously allows to directly express computational secrecy. It should also be relatively easy to extend their result for asymmetric encryption and hash functions (instantiated as random oracles).

7.4 Conclusion and perspectives

In this chapter we summarized our contributions in the area of computational soundness. Most of our work is related to design a general framework to reason about the soundness of the implementation of symbolic models where messages are modelled as a term algebra with an underlying equational theory.

While soundness in the presence of a passive or adaptive adversary is rather well understood the state-of-the-art in the presence of an active adversary is much less advanced. Most results are obtained for a specific protocol language and the definitions are generally tailored to a specific equational theory with monolithic soundness proofs. Our aim is to design a modular framework to show soundness in the presence of an active adversary. This is currently work in progress with H. Comon-Lundh and J.-K. Tsay.

The aim is to define notions of $\vdash_{\mathcal{E}}$-act-soundness and $\sim_{\mathcal{E}}$-act-soundness as cryptographic games, in the style of the soundness definitions for passive and adaptive adversaries. Such games will be independent of a particular protocol language avoiding unnecessary details of a given protocol language. $\vdash_{\mathcal{E}}$-act-soundness can then be used to prove trace mapping results easily for any reasonable protocol specification language, hence avoiding to reprove the soundness results "from scratch" for each language. Similar, $\sim_{\mathcal{E}}$-act-soundness will imply a notion of tree soundness introduced in [CC08] to show observational equivalence. Our preliminary results indicate that we can also get rid of the *unique parsing* assumption which underlies an impossibility result for soundness of exclusive or in the reactive simulatability framework [BP05a]. We also investigate combination results allowing to obtain soundness results for equational theory $\mathcal{E}_1 \cup \mathcal{E}_2$ from soundness results $\mathcal{E}_1$ and $\mathcal{E}_2$.

CHAPTER 8

Perspectives

In this thesis I have summarized a selection of results I obtained in the last few years. I now describe some directions for future work. Parts of these perspectives for future work were already sketched in the conclusions of previous chapters. These research directions will be organized around 4 themes: security APIs, equivalence properties, computational soundness and privacy properties.

8.1 A theory for security APIs

Hardware security modules are designed to allow untrusted code to be executed without compromising the security of key material. Keys are supposed to appear in the clear only in the device's shielded memory. Functionalities of the device, such as encrypting or signing data, are available through an API which generally allows to refer to keys using handles without knowing the keys' values. API level attacks [LR92, BA01] aim at breaking the security policy of such devices, *e.g.* extracting the value of a secret key, by a sequence of calls to the API, which was not foreseen by the designers. Most existing work on formally analyzing security APIs relied on the experience of the use of formal methods in the analysis of security protocols. While security protocols are similar to APIs, there are some important differences which motivate the development of a dedicated theory for analysing APIs. We give here some concrete topics that we plan to explore. This line of research is the PhD topic of Robert Künnemann who started his PhD in October 2010.

Models with storage. As already discussed in Chapter 4 one difference with most security protocols is that security APIs rely on the storage of some data which is persistent. These data can be read or modified by different calls to the API, and affect the execution of API commands. This notion of state was identified by Herzog [Her06] as a major barrier to the application of security protocol analysis tools to verify APIs. We can distinguish different kinds of such state. In some applications we need an unbounded number of stores which take a value chosen from a finite domain. This is for instance the case in PKCS#11 where a particular combination of attributes is associated to each key (and the

number of keys is a priori unbounded). In optimistic fair exchange protocols a similar situation arrives and the trusted third party needs to associate to each session identifier its status (unseen, aborted or resolved). Analyzing protocols which maintain this kind of state was the topic of a recent paper by Mödersheim [Möd10]. However, first experiences with Mödersheim's tool showed that the abstractions he proposes are too coarse to analyze PKCS#11. In other APIs a different kind of state may arise: the TPM for instance has Platform Configuration Registers (PCRs) which can be extended using arbitrary messages. For this kind of applications we need a model that allows a finite number of data stores, which may however take an unbounded number of possible values.

One of our aims is to develop models and analysis techniques which allow us to consider data stores which can be read and updated by parallel processes accessing it. This line of research includes extending cryptographic process calculi with constructs for state. While such state could be encoded in existing process calculi, having dedicated constructs and corresponding proof techniques would be more convenient. Another direction is to revisit decidability and complexity results for models with state when the number of sessions and fresh names is bounded (which may imply that the number of stores or possible values is bounded as well). We also plan to design abstractions which allow to analyse an unbounded number of sessions, as it has been done successfully in the case of "standard" protocols using for instance Horn clauses. A first concrete direction would be to refine Mödersheim's abstraction tool [Möd10] to be able to treat examples such as PKCS#11.

Security properties as ideal systems. As we have noticed in Chapter 4 it is sometimes difficult to define what is the expected security property that a security API should ensure. Therefore, an appealing approach would be to specify the security of APIs by the means of an ideal API. One direction would be to use the framework of universal composability (UC) [Can01]. UC has emerged during the last years as a powerful model for showing the security of complex systems. The main idea is to start with a system which is trivially secure by construction and then refine each of the components towards a realization, generally relying on cryptography. The individual components are specified by an interface and it seems natural to use similar ideas in the context of security APIs. We could also build on our recent work on UC in the applied pi calculus [24]. In particular such an ideal version would ensure that key management and key usage do not interfere. For instance, in PKCS#11 encryption and decryption of data do interfere with the wrap and unwrap constructs for exporting and importing keys which is one of its main sources of flaws. A concrete objective would be to show that APIs such as the ones described in [CS09] and [CC09], which have been shown secure in different models, could implement such a secure functionality.

Computational soundness. During recent years there has been a drive to relate symbolic security proofs and computational security proofs (see [7] for a survey). However, the hypotheses which are needed seem to be unrealistic in the context of security APIs. For instance, the hardware module underlying the API may have restricted computational power and needs to execute efficient encryption algorithms that may not satisfy strong security assumptions that are needed for existing soundness results. We wish to investigate how these hypotheses can be weakened in the particular context of security APIs, exploiting

the particular structure of the data which is allowed in the requests to the API. This work may also built on recent developments in cryptography, *e.g.* deterministic authenticated encryption [RS06], allowing for instance to securely encrypt keys with their attributes.

8.2 From trace properties to indistinguishability properties

The notion of indistinguishability, such as static equivalence, observational equivalence or testing equivalence, has shown to be extremely useful to model security properties. As we have seen in Chapter 3, privacy-type properties of electronic voting can be expressed in terms of observational equivalence. Also, strong notions of secrecy [Bla04], resistance to guessing attacks [CDE05, Bau05, 27] or more generally *real-or-random* properties can be modelled using indistinguishability. Indistinguishability of processes can also be used to express security by the means of ideal systems [AG99, 24].

While we have a good understanding of trace properties, such as deduction based secrecy and authentication, the situation is different for indistinguishability properties. The picture for static equivalence is also rather complete [AC06, CD07, ACD07, BBRC09] and exact [BCD09, 23, 6] and approximate [BAF05] tools exist for large classes of equational theories. The situation is however less clear when we consider indistinguishability in the presence of an active adversary. Current results allow to approximate observational equivalence for an unbounded [BAF05], respectively bounded number of sessions [31, 9], or decide it on a restricted class of processes [CD09a], relying on decision procedure for deciding the equivalence of two symbolic traces [Bau05, CR10], for subterm convergent equational theories.

Decidability and complexity. Obviously, when we allow for replication, equivalence properties become undecidable. As shown by Hüttel [Hüt02], this is even the case for the finite control fragment, for which observational equivalence is decidable in the original pi calculus. It may however be possible to identify subclasses of protocols for which the problem becomes decidable, as for trace properties [DEK82, RS03, RS05, Low98, 26]. Even though, one may expect observational and testing equivalence to be decidable for many classes of equational theories for the finite applied pi calculus (*i.e.* the fragment without replication) such results are currently missing and our theoretical understanding of this decision problem is currently very limited. Therefore we foresee to investigate decidability and complexity results for different (families of) equational theories.

One may note that equivalence of constraint systems such as [Bau07, CCD10, CR10], allows us to decide the equivalence of two symbolic traces. While this is an interesting first step towards the decision of equivalence properties it is not sufficient for deciding testing equivalence or observational equivalence (except for restricted classes of processes [CD09a] for which the problem has been shown to be co-NP complete). (These results nevertheless allow us to approximate observational equivalence using for instance our symbolic bisimulations [31, 9].)

Combination. In order to develop decision procedure in a modular way it would be interesting to obtain combination results of the following type: if observational equivalence is decidable for the equational theory $\mathcal{E}_1$ and for the equational theory $\mathcal{E}_2$ then observational

equivalence is also decidable for $\mathcal{E}_1 \cup \mathcal{E}_2$. Such results have been obtained for disjoint theories in the case of static equivalence [ACD07], as well as for disjoint theories [CR05] and hierarchical theories [CR06] in the case of reachability properties. To the best of our knowledge no such combination results exist for equivalence properties in the presence of an active attacker.

Efficient procedures. While a theoretical understanding of the decidability and complexity is important and interesting, it is equally important to develop practical tools that can be used to analyse and verify protocols. The Proverif tool indeed allows to approximate observational equivalence [BAF05] by showing a finer relation. However, this relation is too fine in examples such as the electronic voting protocols we studied in [37, 34, 13]. Also, the procedures for deciding equivalence of two constraint systems presented in [Bau07, CCD10] are non-deterministic and not suitable for an implementation. In [CCD10] a procedure and and a prototype implementation are presented, but only for a fixed signature. With Ş. Ciobâcă and R. Chadha we are currently working on a generalization of the procedure implemented in KiSs to decide trace equivalence for the finite applied pi calculus and subterm equational theories.

Composition. Currently, most works on composition are limited to trace properties [CDD07, CD09b, CC10, 26]. It would be interesting to investigate whether these results can be transferred to indistinguishability properties. The aim is to find sufficient conditions such that

$$\textbf{if} \quad \nu k.P \sim \nu k.Q \quad \textbf{then} \quad \nu k.(P \mid R) \sim \nu k.(Q \mid R)$$

and similarly for self-composition

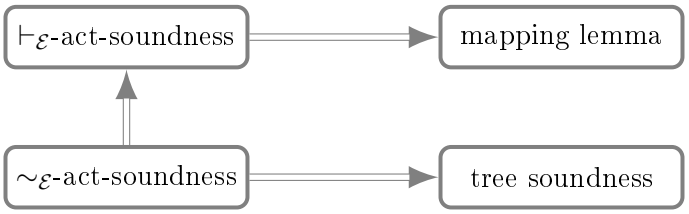
$$\textbf{if} \quad \nu k.P \sim Q \quad \textbf{then} \quad \nu k.!P \sim !Q$$

The proofs in [CD09b, 26] are based on a constraint solving procedure for deciding security properties. In recent work, Cheval *et al.* [CCD10] also use a constraint solving procedure to decide equivalence properties. A concrete question is whether the proofs of [CD09b, 26] can be adapted to this procedure to obtain composition results for indistinguishability properties.

8.3 Modular computational soundness results

A first computational soundness result was obtained by Abadi and Rogaway [AR02]. Subsequent work extended the adversary capacities, the cryptographic primitives which are considered and the properties. While soundness in the presence of a passive or adaptive adversary is rather well understood the state-of-the-art in the presence of an active adversary is much less advanced. Most results are obtained for a specific protocol language and the definitions are generally tailored to a specific equational theory with monolithic soundness proofs. Our aim is to design a modular framework to show soundness in the presence of an active adversary. This is currently work in progress with H. Comon-Lundh and J.-K. Tsay.

The aim is to define notions of $\vdash_{\mathcal{E}}$-act-soundness and $\sim_{\mathcal{E}}$-act-soundness as cryptographic games, in the style of the soundness definitions for passive and adaptive adversaries. One of the main differences with the games for passive and adaptive adversaries is that the attacker may himself compute and submit bitstrings to the oracles, rather than only symbolic terms. Such games will be independent of a particular protocol language avoiding unnecessary details of a given protocol language and allow us to concentrate on the hypotheses we need on the cryptographic primitives. $\vdash_{\mathcal{E}}$-act-soundness can then be used to prove trace mapping results easily for any reasonable protocol specification language, hence avoiding to reprove the soundness results "from scratch" for each language. Similarly, $\sim_{\mathcal{E}}$-act-soundness will imply a notion of tree soundness introduced in [CC08] to show observational equivalence. Our preliminary results indicate that we can also get rid of the *unique parsing* assumption which underlies an impossibility result for soundness of exclusive or in the reactive simulatability framework [BP05a]. Hence, we hope to get results that can be summarized as follows:



We also investigate combination results allowing to obtain soundness results for equational theory $\mathcal{E}_1 \cup \mathcal{E}_2$ from soundness results $\mathcal{E}_1$ and $\mathcal{E}_2$. While we cannot expect to achieve such results in general we aim at identifying sufficient conditions for such combination results to hold.

Another direction for research in the area of computational soundness is to make soundness results more applicable to real case studies. Currently, many simplifying hypotheses are made to allow such soundness proofs. For instance, encryption schemes are supposed to be length concealing and adversaries are not allowed to generate keys in an arbitrary way which is not realistic in practice. Hence, many of these results need to be refined to be applicable. We believe that giving a computational proof of the anonymity of a simple voting protocol such as [FOO92] using a soundness result would be a challenging task which would highlighten many shortcomings in the current results. Another application area which we already mentioned in the previous section and which certainly needs to adapt existing results are security APIs. When trying to apply such soundness results we may also need to sometimes refine symbolic models and adapt the symbolic analysis techniques.

8.4 Privacy properties

Strong notions of privacy. In our previous work on electronic voting protocols [34, 13] we have studied strong notions of privacy, such as receipt-freeness and coercion resistance. The aim of these properties is to avoid that a voter can break privacy himself, *i.e.*, the voter should not be able to prove how he voted to a third party. While these properties are natural in the context of electronic voting we believe that similar notions of strong

privacy also arise in other contexts. Receipt-freeness has for instance been identified as a desirable property in electronic auction protocols [AS02] to avoid bid-rigging and this property has recently been modelled and analysed [DJP10] using definitions based on our work in the context of electronic voting. Such strong notions of privacy can certainly be applied in other areas. For instance, Backes *et al.* [BMM10] recently proposed symbolic abstractions for secure multi-party computations. We plan to investigate formalization of strong notions of privacy in the general case of secure multi-party computations.

We are also planning to extend our epistemic logic [22] to be able to express notions such as receipt-freeness and coercion-resistance. In particular we need to introduce the notion agents and their knowledge. We expect to be able to express Küsters *et al.*'s epistemic definition of coercion-resistance [KT09] in this logic and maybe compare it to definitions in terms of indistinguishability.

Privacy and mobility. The proliferation of mobile devices, such as mobile smart phones, RFID tags, ..., and ad-hoc networks has lead to a wide range of security problems. In particular, privacy properties, such as unlinkability, are of increasing importance and difficult to ensure. Recently, the use of RFID tags in passports has been formally studied [ACRR10]. This analysis was conducted in the applied pi calculus. The applied pi calculus is however limited as it does not have particular constructs for mobility. While specialized calculi with constructs for modelling mobility exist [CG00, NH06], they do not have constructs for modelling cryptographic primitives. An interesting direction for research would be to enhance such calculi with a term algebra as in the applied pi calculus to be able to reason explicitly about security properties in such mobile environments. Some works in this direction already exist, such as Blanchet and Aziz's crypto-loc calculus [BA03] and higher order cryptographic pi calculi [KH11, SS09] for modelling mobile code.

Bibliography

[ABB+05] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification (CAV'05)*, Lecture Notes in Computer Science, pages 281–285. Springer, 2005.

[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In *Advances in Cryptology - EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 403–422. Springer, 2010.

[ABF07] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):1–59, 2007.

[ABW10] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. *Journal of Computer Security*, 18(5):909–968, 2010.

[AC06] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.

[ACC+08] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *Proc. 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10, 2008.

[ACD07] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Combining algorithms for deciding knowledge in security protocols. In *Proc. 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117. Springer, 2007.

[ACRR10] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

[AD07] Myrto Arapinis and Marie Duflot. Bounding messages for free in security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 376–387. Springer, 2007.

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, MIT, 2006.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *Proc. 17th Usenix Security Symposium*, pages 335–348. USENIX Association, 2008.

[AdMPQ09] Ben Adida, Olivier de Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Electronic Voting Technology/Workshop on Trustworthy Elections (EVT/WOTE)*, 2009.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proc. 28th ACM Symp. on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004.

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[AL00] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. of the 12th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 380–394, 2000.

[ALV02a] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[ALV02b] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290:695–740, 2002.

[AN95] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In *Advances in Cryptology – Crypto 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 1995.

[AN06] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. In *Proc. IEEE Symposium on Security and Privacy (SP'94)*, pages 122–136. IEEE Comp. Soc. Press, 2006.

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

[Ara08] Myrto Arapinis. *Sécurité des protocoles cryptographiques : décidabilité et résultats de réduction*. Thèse de doctorat, Université Paris 12, Créteil, France, November 2008.

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Tripartite authenticated key agreement protocols from pairings. In *Proc. 9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 332–359. Springer, 2003.

[AS02] Masayuki Abe and Koutarou Suzuki. Receipt-free sealed-bid auction. In *Proc. 5th International Conference on Information Security (ISC'02)*, volume 2433 of *Lecture Notes in Computer Science*, pages 191–199. Springer, 2002.

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proc. 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 8–17. ACM Press, April 1997.

[AVI] AVISPA Collection of Security Protocols. `http://www.avispa-project.org/library/`.

[AW05] Martín Abadi and Bogdan Warinschi. Password-based encryption analyzed. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 664–676. Springer, 2005.

[BA01] M. Bond and R. Anderson. API level attacks on embedded systems. *IEEE Computer Magazine*, pages 67–75, October 2001.

[BA03] Bruno Blanchet and Benjamin Aziz. A calculus for secure mobility. In *Proc. 8th Asian Computing Science Conference (ASIAN'03)*, volume 2896 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2003.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Proc. 20th Symposium on Logic in Computer Science (LICS'05)*, pages 331–340. IEEE Comp. Soc. Press, 2005.

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal London Society. Series A. Mathematical and Physical Sciences*, 426(1871):233–271, 1989.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[Bau07] Mathieu Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, LSV, ENS Cachan, France, 2007.

[Bau08] Mathieu Baudet. YAPA (Yet Another Protocol Analyzer), 2008. `http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.

[BBN04] Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic bisimulation in the spi calculus. In *Proc. 15th Int. Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.

[BBRC09] Mouhebeddine Berrima, Narjes Ben Rajeb, and Véronique Cortier. Deciding knowledge in security protocols under some e-voting theories. Research Report RR-6903, INRIA, 2009.

[BC08] Bruno Blanchet and Avik Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *Proc. Symposium on Security and Privacy (SP'08)*, pages 417–431, 2008.

[BCD09] Mathieu Baudet, Véronique Cortier, and Stéphanie Delaune. YAPA: A generic tool for computing intruder knowledge. In *Proc. 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163, Brasília, Brazil, 2009. Springer.

[BCFS10] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing PKCS#11 security tokens. In *Proc. 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 260–269. ACM Press, 2010.

[BCP01] E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In *Advances in Cryptology - ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–309. Springer, 2001.

[BD94] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system (extended abstract). In *Advances in Cryptology - EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1994.

[BDNP99] M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Proc. 14th Symposium on Logic in Computer Science (LICS'99)*, pages 157–166. IEEE Comp. Soc. Press, 1999.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology (CRYPTO'01)*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BHM08] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proc. 21st IEEE Computer Security Foundations Symposium, (CSF'08)*, pages 195–209. IEEE Comp. Soc. Press, 2008.

[BJ03] Michael Backes and Christian Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *Proc. 20th Symposium on Theoretical Aspects of Computer Science (STACS'03)*, pages 675–686. Springer, 2003.

[Bla01] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[Bla04] Bruno Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *Proc. Symposium on Security and Privacy (SP'04)*, pages 86–100. IEEE Comp. Soc. Press, 2004.

[Bla06] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *Proc. Symposium on Security and Privacy (SP'06)*, pages 140–154. IEEE Comp. Soc. Press, 2006.

[Bla08] Bruno Blanchet. *Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire. Automatic verification of security protocols: formal model and computational model.* Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, November 2008.

[Bla09] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.

[BLMW07] Emmanuel Bresson, Yassine Lakhnech, Laurent Mazaré, and Bogdan Warinschi. A generalization of ddh with applications to protocol analysis and computational soundness. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2007.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proc. Symposium on Security and Privacy (SP'92)*, pages 72–84. IEEE Comp. Soc. Press, 1992.

[BMM10] Michael Backes, Matteo Maffei, and Esfandiar Mohammadi. Computationally sound abstraction and verification of secure multi-party computations. In *Proc. 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'10)*, volume 8 of *Leibniz International Proceedings in Informatics*, pages 352–363. Leibniz-Zentrum für Informatik, 2010.

[BN96] Michele Boreale and Rocco De Nicola. A symbolic semantics for the pi-calculus. *Information and Computation*, 126(1):34–52, 1996.

[BN02] Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. In *Proc. 9th International Conference on Algebraic Methodology and Software Technology (AMAST'02)*, volume 2422 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2002.

[Bon01] M. Bond. Attacks on cryptoprocessor transaction sets. In *Proc. 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*, volume 2162 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 2001.

[Bor01] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of the 28th Int. Coll. Automata, Languages, and Programming (ICALP'01)*, volume 2076, pages 667–681. Springer, 2001.

[Bor08] Johannes Borgström. *Equivalences and Calculi for Formal Verifiation of Cryptographic Protocols*. Phd thesis, EPFL, Switzerland, 2008.

[BP03] Bruno Blanchet and Andreas Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proc. Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2003.

[BP05a] Michael Backes and Birgit Pfitzmann. Limits of the cryptographic realization of Dolev-Yao-style xor. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 336–354, 2005.

[BP05b] Michael Backes and Birgit Pfitzmann. Relating cryptographic und symbolic key secrecy. In *Proc. 26th IEEE Symposium on Security and Privacy (SP'05)*, pages 171–182, 2005.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003.

[BPW07] Michael Backes, Birgit Pfitzmann, and Michael Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

[BRS07] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proc. 11th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 62–71, 2007.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th Symposium on Theory of Computing (STOC'94)*, pages 544–553. ACM Press, 1994.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Moni Naor, editor, *Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS'01)*, pages 136–145. IEEE Comp. Soc. Press, 2001.

[CC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *Proc. 15th ACM Conference on Computer and Communications Security (CCS'08)*, pages 109–118. ACM Press, 2008.

[CC09] C. Cachin and N. Chandran. A secure cryptographic token interface. In *Proc. 22th Computer Security Foundations Symposium (CSF'09)*, pages 141–153. IEEE Comp. Soc. Press, 2009.

[CC10] Ştefan Ciobâcă and Véronique Cortier. Protocol composition for arbitrary primitives. In *Proc. 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 322–336. IEEE Comp. Soc. Press, 2010.

[CCD10] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Automating security analysis: symbolic equivalence of constraint systems. In *Proc. 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426. Springer-Verlag, 2010.

[CCG+02] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV'02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.

[CCK+07] Ran Canetti, Ling Cheung, Dilsun Kaynar, Nancy Lynch, and Olivier Pereira. Compositional security for Task-PIOAs. In *Proc. 20th Computer Security Foundations Symposium (CSF'07)*, pages 125–139. IEEE Comp. Soc. Press, 2007.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a secure voting system. In *Proc. Symposium on Security and Privacy (SP'08)*, pages 354–368, Washington, DC, USA, 2008. IEEE Comp. Soc. Press.

[CCZ10] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zălinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic*, 11(2), January 2010.

[CD05] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proc. of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

[CD07] Véronique Cortier and Stéphanie Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Proc. 14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *LNAI*, pages 196–210. Springer, October 2007.

[CD09a] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Proc. 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Comp. Soc. Press, 2009.

[CD09b] Véronique Cortier and Stéphanie Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, February 2009.

[CDD07] Véronique Cortier, Jérémie Delaitre, and Stéphanie Delaune. Safely composing security protocols. In V. Arvind and Sanjiva Prasad, editors, *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 352–363. Springer, December 2007.

[CDE05] Ricardo Corin, Jeroen Doumen, and Sandro Etalle. Analysing password protocol security against off-line dictionary attacks. *Electronic Notes in Theoretical Computer Science*, 121:47–63, 2005.

[CDS07] V. Cortier, S. Delaune, and G. Steel. A formal theory of key conjuring. In *Proc. 20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 79–93. IEEE Comp. Soc. Press, 2007.

[CE03] Ricardo Corin and Sandro Etalle. An improved constraint-based system for the verification of security protocols. In *Static Analysis Symposium—SAS 2002*, volume 2477 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2003.

[CES06] Ricardo Corin, Sandro Etalle, and Ari Saptawijaya. A logic for constraint-based security protocol analysis. In *Proc. IEEE Symposium on Security and Privacy (SP'06)*, pages 155–168. IEEE Comp. Soc. Press, 2006.

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation (extended abstract). In *Proc. 37th Symposium on Foundations of Computer Science (FOCS'96)*, pages 504–513. IEEE Comp. Soc. Press, 1996.

[CG00] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.

[CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols (extended abstract). In *Proc. 3rd Theory of Cryptography Conference (TCC'06)*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.

[Cio09] Ştefan Ciobâcă. KiSs, 2009. `http://www.lsv.ens-cachan.fr/~ciobaca/kiss`.

[CJ97] J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps.gz`, 1997.

[CJP07] Hubert Comon-Lundh, Florent Jacquemard, and Nicolas Perrin. Tree automata with memory, visibility and structural constraints. In *Proc. 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'07)*, volume 4423 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2007.

[CKRT03a] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with diffie-hellman exponentiation and products in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2003.

[CKRT03b] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An np decision procedure for protocol insecurity with XOR. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 261–270. IEEE Comp. Soc. Press, 2003.

[CKS07] V. Cortier, G. Keighren, and G. Steel. Automatic analysis of the security of xor-based key management schemes. In *Proc. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *Lecture Notes in Computer Science*, pages 538–552. Springer, 2007.

[CLC05] H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1):143–214, February 2005.

[Clu03] J. Clulow. On the security of PKCS#11. In *Proc. 5th International Worshop on Cryptographic Hardware and Embedded Systems (CHES'03)*, volume 2779 of *LNCS*, pages 411–425. Springer, 2003.

[CM06] J. Courant and J.-F. Monin. Defending the bank with a proof assistant. In *Proc. 6th International Workshop on Issues in the Theory of Security (WITS'06)*, pages 87 – 98, 2006.

[Com94] Hubert Comon. Inductionless induction. In *2nd International Conference in Logic For Computer Science: Automated Deduction. Lecture notes.* Université de Savoie, 1994.

[Cor02] V. Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. application to cryptographic protocols analysis. In *Research Report LSV-02-3*, March 2002.

[CR05] Yannick Chevalier and Michael Rusinowitch. Combining intruder theories. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651. Springer, 2005.

[CR06] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In *Proc. 17th International Conference on Rewriting Techniques and Applications (RTA'06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2006.

[CR10] Yannick Chevalier and Michaël Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 2010. To appear.

[Cre08a] Cas J.F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. 20th International Conference on Computer Aided Verification (CAV'08)*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[Cre08b] Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *Proc. 15th Conference on Computer and Communications Security (CCS'08)*, pages 119–128. ACM Press, 2008.

[CS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 271–280. IEEE Comp. Soc. Press, June 2003.

[CS09] Véronique Cortier and Graham Steel. A generic security API for symmetric key management on cryptographic devices. In *Proc. 14th European Symposium on Research in Computer Security (ESORICS'09)*, volume 5789 of *Lecture Notes in Computer Science*, pages 605–620. Springer, 2009.

[CW05] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[DC] Roberto Di Cosmo. On privacy and anonymity in electronic and non electronic voting: the ballot-as-signature attack.

[DDMR07] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (pcl). *Electronic Notes in Theoretical Computer Science*, 172:311–358, 2007.

[DDS10] Morten Dahl, Stéphanie Delaune, and Graham Steel. Formal analysis of privacy for vehicular mix-zones. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 2010.

[DEK82] D. Dolev, S. Even, and R. M. Karp. On the security of ping-pong protocols. In *Proc. Advances in Cryptology - CRYPTO'82*, pages 177–186, 1982.

[DGT07] Shaddin F. Doghmi, Joshua D. Guttman, and F. Javier Thayer. Skeletons, homomorphisms, and shapes: Characterizing protocol executions. *Electronic Notes in Theoretical Computer Science*, 173:85–102, 2007.

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

[Dij75] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975.

[DJP10] Naipeng Dong, Hugo Jonker, and Jun Pang. Analysis of a receipt-free auction protocol in the applied pi calculus. In Sandro Etalle and Joshua Guttman, editors, *Proc. International Workshop on Formal Aspects in Security and Trust (FAST'10)*, Pisa, Italy, 2010. To appear.

[DLM04] Nancy A. Durgin, Patrick Lincoln, and John C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.

[DY81] D. Dolev and A.C. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of Computer Science*, pages 350–357. IEEE Comp. Soc. Press, 1981.

[EG83] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *Proc. 24th Symposium on Foundations of Computer Science (FOCS'83)*, pages 34–39. IEEE Comp. Soc. Press, 1983.

[EMM09] Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-npa: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2009.

[FHMV95] Ronald Fagin, Joseph Yoram Halpern, Y. Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[FM99] Riccardo Focardi and Fabio Martinelli. A uniform approach for the definition of security properties. In *Proc. of World Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 794–813. Springer, 1999.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazui Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[FS09] S. Fröschle and G. Steel. Analysing PKCS#11 key management APIs with unbounded fresh data. In *Proc. Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'09)*, volume 5511 of *Lecture Notes in Computer Science*, pages 92–106, York, UK, 2009. Springer.

[GK00] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification. In *Proc. 17th International Conference on Automated Deduction (CADE'00)*, volume 1831 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2000.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proc. 14th Annual ACM Symposium on Theory of Computing (STOC'82)*. ACM Press, 1982.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on Theory of Computing (STOC'87)*, pages 218–229. ACM press, 1987.

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. Symposium on Security and Privacy (SP'90)*, pages 234–248. IEEE Comp. Soc. Press, 1990.

[Gon93] Li Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in Communications*, 11(5):657–662, 1993.

[Gou05] Jean Goubault-Larrecq. Deciding $\mathcal{H}_1$ by resolution. *Information Processing Letters*, 95(3):401–408, August 2005.

[Gou08] Jean Goubault-Larrecq. Towards producing formally checkable security proofs, automatically. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 224–238. IEEE Comp. Soc. Press, 2008.

[GRS+07] Sigrid Gürgens, Carsten Rudolph, Dirk Scheuermann, Marion Atts, and Rainer Plaga. Security evaluation of scenarios based on the TCG's TPM specification. In *Proc. 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2007.

[GS95] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Dependable Computing for Critical Applications 5*, pages 44–55. IEEE Comp. Soc. Press, 1995.

[GS05] Prateek Gupta and Vitaly Shmatikov. Towards computationally sound symbolic analysis of key exchange protocols. In *Proc. workshop on Formal methods in security engineering (FMSE'05)*, pages 23–32. ACM Press, 2005.

[GT00a] Joshua Guttman and F. Javier Thayer Fabrega. Protocol independence through disjoint encryption. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

[GT00b] Joshua D. Guttman and F. Javier Thayer. Authentication tests. In *Proc. Symposium on Security and Privacy (SP'00)*, pages 96–109. IEEE Comp. Soc. Press, 2000.

[GT00c] Joshua D. Guttman and F. Javier Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

[Gut01] Joshua D. Guttman. *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, chapter Security Goals: Packet Trajectories and Strand Spaces, pages 197–261. Springer-Verlag, 2001.

[Her06] J. Herzog. Applying protocol analysis to security device interfaces. *IEEE Security & Privacy Magazine*, 4(4):84–87, July-Aug 2006.

[HL95] Matthew Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[HO05] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[HS00] J. Heather and S. Schneider. Towards automatic verification of authentication protocols on an unbounded network. In *Proc. of the 13th Computer Security Foundations Workshop (CSFW'00)*, pages 132–143. IEEE Comp. Soc. Press, 2000.

[Hüt02] Hans Hüttel. Deciding framed bisimilarity. In *Proc. 4th International Workshop on Verification of Infinite-State Systems (INFINITY'02)*, pages 1–20, 2002.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proc. Workshop on Privacy in the Electronic Society (WPES'05)*. ACM Press, 2005.

[JLM06] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Computational soundness of symbolic analysis for protocols using hash functions. In *Proceedings of the Workshop on Information and Computer Security (ICS'06)*, Electronic Notes in Theoretical Computer Science, Timisoara, Romania, September 2006. Elsevier Science Publishers.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *Proc. 4th International Symposium on Algorithmic Number Theory (ANTS-IV)*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[JVP09] Magnus Johansson, Björn Victor, and Joachim Parrow. A fully abstract symbolic semantics for psi-calculi. In *Proc. 6th Workshop on Structural Operational Semantics (SOS'09)*, volume 18 of *Electronic Proceedings in Theoretical Computer Science*, pages 17–31, 2009.

[KDMR08] Ralf Küsters, Anupam Datta, John C. Mitchell, and Ajith Ramanathan. On the relationships between notions of simulation-based security. *Journal of Cryptology*, 21(4):492–546, 2008.

[KH11] Vasileios Koutavas and Matthew Hennessy. A testing theory for a higher-order cryptographic language (extended abstract). In *Proc. 20th European Symposium on Programming (ESOP'11)*, Lecture Notes in Computer Science. Springer, 2011. To appear.

[Kri10] Manuel J. Kripp. Three internet elections in europe in 2011. *Modern democracy - The Electronic Voting and Participation Magazine*, 2:11, November 2010.

[KT08] Ralf Küsters and Max Tuengerthal. Joint state theorems for public-key encryption and digitial signature functionalities with local computation. In *Proc. 21st IEEE Computer Security Foundations Symposium (CSF'08)*. IEEE Comp. Soc. Press, 2008.

[KT09] Ralf Küsters and Tomasz Truderung. An Epistemic Approach to Coercion-Resistance for Electronic Voting Protocols. In *Proc. Symposium on Security and Privacy (SP'09)*, pages 251–266. IEEE Comp. Soc. Press, 2009.

[KTV10] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. A game-based definition of coercion-resistance and its applications. In *Proc. 23rd Computer Security Foundations Symposium (CSF'10)*, pages 122–136. IEEE Comp. Soc. Press, 2010.

[Küs06] Ralf Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proc. 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 309–320. IEEE Comp. Soc. Press, 2006.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proc. Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer, 2004.

[LL10] Jia Liu and Huimin Lin. A complete symbolic bisimulation for full applied pi calculus. In *Proc. 36th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'10)*, volume 5901 of *Lecture Notes in Computer Science*, pages 552–563. Springer, 2010.

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

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis OF Systems—TACAS 1996*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[Low97] Gavin Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. 10th Computer Security Foundations Workshop (CSFW'97)*, pages 18–30. IEEE Comp. Soc. Press, 1997.

[Low98] Gavin Lowe. Towards a completeness result for model checking of security protocols. In *Proc. 11th Computer Security Foundations Workshop (CSFW'98)*, pages 96–106. IEEE Comp. Soc. Press, 1998.

[Low99] Gavin Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.

[LR92] D. Longley and S. Rigby. An automatic search for security flaws in key management schemes. *Computers and Security*, 11(1):75–89, March 1992.

[Mea96] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[Mer09] Antoine Mercier. *Contributions à l'analyse automatique des protocoles cryptographiques en présence de propriétés algébriques : protocoles de groupe, équivalence statique.* Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, December 2009.

[Mit02] John C. Mitchell. Multiset rewriting and security protocol analysis. In *Proc. 13th International Conference on Rewriting Techniques and Applications (RTA'02)*, volume 2378 of *Lecture Notes in Computer Science*, pages 19–22. Springer, 2002.

[MN06] T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *Advances in Cryptology - CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.

[Möd10] Sebastian Mödersheim. Abstraction by set-membership: verifying security protocols and web services with databases. In *Proc. 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 351–360. ACM Press, 2010.

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

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th Conference on Computer and Communications Security*, pages 166–175. ACM Press, 2001.

[MW04] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[NH06] Sebastian Nanz and Chris Hankin. Formal security analysis for ad-hoc networks. *Electronic Notes in Theoretical Computer Science*, 142:195–213, 2006.

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

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *Proc. IFIP World Conference on IT Tools*, pages 21–30, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35, Paris, France, 1997. Springer.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1/2):85–128, 1998.

[PP04] Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Proc. Selected Areas in Cryptography (SAC'04)*, volume 3357 of *Lecture Notes in Computer Science*, pages 185–200, 2004.

[RDD+08] Arnab Roy, Anupam Datta, Ante Derek, John C. Mitchell, and Jean-Pierre Seifert. *Formal Logical Methods for System Security and Correctness*, volume 14 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, chapter Secrecy Analysis in Protocol Composition Logic, pages 199 – 232. IOS Press, 2008.

[RS03] Ramaswamy Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *LNCS*, pages 363–374. Springer, 2003.

[RS05] Ramaswamy Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[RS06] P. Rogaway and T. Shrimpton. Deterministic authenticated encryption: A provable-security treatment of the keywrap problem. In *Advances in Cryptology - EUROCRYPT'06*, volume 4004 of *LNCS*. Springer, 2006.

[RSA04] RSA Security Inc., v2.20. *PKCS #11: Cryptographic Token Interface Standard.*, June 2004.

[RSG+00] P.Y.A Ryan, S.A. Schneider, M. Goldsmith, G. Lowe, and A.W. Roscoe. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Comp. Soc. Press, 2001.

[Sar] Luis Sarmenta. TPM/J developer's guide. Massachussetts Institute of Technology.

[SC10] Ben Smyth and Véronique Cortier. Does helios ensure ballot secrecy? Cryptology ePrint Archive, Report 2010/625, 2010.

[Shm04] Vitaly Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *European Symposium on Programming (ESOP'04)*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369. Springer, 2004.

[SPO] Spore: Security protocols open repository. `http://www.lsv.ens-cachan.fr/Software/spore/`.

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proc. 4th European Symposium On Research In Computer Security (ESORICS'96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer, 1996.

[SS09] Nobuyuki Sato and Eijiro Sumii. The higher-order, call-by-value applied pi-calculus. In *Proc. 7th Asian Symposium on Programming Languages and Systems (APLAS'09)*, volume 5904 of *Lecture Notes in Computer Science*, pages 311–326. Springer, 2009.

[Ste05] G. Steel. Deduction with XOR constraints in security API modelling. In *Proc. 20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Computer Science*, pages 322–336, Tallinn, Estonia, 2005. Springer.

[SvO94] Paul Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *Proc. Symposium on Security and Privacy (SP'94)*, pages 14–28. IEEE Comp. Soc. Press, 1994.

[Syv96] Paul F. Syverson. Limitations on design principles for public key protocols. In *Proc. Symposium on Security and Privacy (SP'96)*, pages 62–72. IEEE Comp. Soc. Press, 1996.

[TCG07] Trusted Computing Group. TPM Specification version 1.2. Parts 1–3, revision 103. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, 2007.

[TEB05] Ferucio Laurentiu Tiplea, Constantin Enea, and Catalin V. Birjoveanu. Decidability and complexity results for security protocols. In *Proc. Verification of Infinite-State Systems with Applications to Security (VISSAS'05)*, volume 1 of *NATO Security through Science Series D: Information and Communication Security*, pages 185–211. IOS Press, 2005.

[THG99] F. Javier Thayer Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[Tur06] Mathieu Turuani. The cl-atse protocol analyser. In *Proc. 17th International Conference on Rewriting Techniques and Applications (RTA'06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Springer, 2006.

[UM10] Dominique Unruh and Jörn Müller-Quade. Universally composable incoercibility. In *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 411–428, 2010.

[Wei99] Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proc. 16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 1999.

[WL92] T.Y.C. Woo and S.S. Lam. Authentication for distributed systems. In *Proc. Symposium on Security and Privacy (SP'92)*, pages 178–194. IEEE Comp. Soc., 1992.

[WL94] T.Y.C. Woo and S.S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 28(3):24–37, 1994.

Publication list

A complete, up-to-date list of my publications with most of them available online is available at

`http://www.lsv.ens-cachan.fr/~kremer/mes_publis.php`

Book chapter

[1] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Mirosław Kutyłowski, and Ben Adida, editors, *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer, May 2010.

Edited books

[2] Michele Boreale and Steve Kremer, editors. *Proceedings of the 7th International Workshop on Security Issues in Concurrency (SecCo'09)*, volume 7 of *Electronic Proceedings in Theoretical Computer Science*, Bologna, Italy, August 2009.

[3] Liqun Chen, Steve Kremer, and Mark D. Ryan, editors. *Formal Protocol Verification Applied*, volume 07421 of *Dagstuhl Seminar Proceedings*, Dagstuhl, Germany, 2008.

[4] Véronique Cortier and Steve Kremer, editors. *Formal Models and Techniques for Analyzing Security Protocols*. Cryptology and Information Security Series. IOS Press, 2011. To appear.

[5] Steve Kremer and Prakash Panangaden, editors. *Proceedings of the 6th International Workshop on Security Issues in Concurrency (SecCo'08)*, volume 242 of *Electronic Notes in Theoretical Computer Science*, Toronto, Canada, August 2009. Elsevier Science Publishers.

Journal publications

[6] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2011, Springer. To appear.

[7] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 2011, Springer. To appear.

[8] Steve Kremer, Antoine Mercier, and Ralf Treinen. Reducing equational theories for the decision of static equivalence. *Journal of Automated Reasoning*, 2011, Springer. To appear.

[9] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, March 2010, IOS Press.

[10] Stéphanie Delaune, Steve Kremer, and Graham Steel. Formal analysis of PKCS#11 and proprietary extensions. *Journal of Computer Security*, 18(6):1211–1245, November 2010, IOS Press.

[11] Steve Kremer and Laurent Mazaré. Computationally sound analysis of protocols using bilinear pairings. *Journal of Computer Security*, 18(6):999–1033, November 2010, IOS Press.

[12] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. *Information and Computation*, 207(4):496–520, April 2009, Elsevier Science Publishers.

[13] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009, IOS Press.

[14] Jean Cardinal, Steve Kremer, and Stefan Langerman. Juggling with pattern matching. *Theory of Computing Systems*, 39(3):425–437, June 2006, Springer.

[15] Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party contract signing. *Journal of Automated Reasoning*, 36(1-2):39–83, January 2006, Springer.

[16] Steve Kremer and Olivier Markowitch. Fair multi-party non-repudiation. *International Journal on Information Security*, 1(4):223–235, July 2003, Springer-Verlag.

[17] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security*, 11(3):399–429, 2003, IOS Press.

[18] Steve Kremer, Olivier Markowitch, and Jianying Zhou. An intensive survey of non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, November 2002, Elsevier.

Conference publications

[19] Stéphanie Delaune, Steve Kremer, Mark Ryan, and Graham Steel. A formal analysis of authentication in the TPM. In Sandro Etalle and Joshua Guttman, editors, *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, Pisa, Italy, September 2010. To appear.

[20] Steve Kremer, Mark D. Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis and Bart Preneel, editors, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404, Athens, Greece, September 2010. Springer.

[21] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In Alessandro Armando and Gavin Lowe, editors, *Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (ARSPA-WITS'10)*, volume 6186 of *Lecture Notes in Computer Science*, pages 146–163, Paphos, Cyprus, October 2010. Springer.

[22] Rohit Chadha, Stéphanie Delaune, and Steve Kremer. Epistemic logic for the applied pi calculus. In David Lee, Antónia Lopes, and Arnd Poetzsch-Heffter, editors, *Proceedings of IFIP International Conference on Formal Techniques for Distributed Systems (FMOODS/FORTE'09)*, volume 5522 of *Lecture Notes in Computer Science*, pages 182–197, Lisbon, Portugal, June 2009. Springer.

[23] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. In Renate Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*, Lecture Notes in Artificial Intelligence, pages 355–370, Montreal, Canada, August 2009. Springer.

[24] Stéphanie Delaune, Steve Kremer, and Olivier Pereira. Simulation based security in the applied pi calculus. In Ravi Kannan and K. Narayan Kumar, editors, *Proceedings of the 29th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, volume 4 of *Leibniz International Proceedings in Informatics*, pages 169–180, Kanpur, India, December 2009. Leibniz-Zentrum für Informatik.

[25] Steve Kremer, Antoine Mercier, and Ralf Treinen. Reducing equational theories for the decision of static equivalence. In Anupam Datta, editor, *Proceedings of the 13th Asian Computing Science Conference (ASIAN'09)*, volume 5913 of *Lecture Notes in Computer Science*, pages 94–108, Seoul, Korea, December 2009. Springer.

[26] Myrto Arapinis, Stéphanie Delaune, and Steve Kremer. From one session to many: Dynamic tags for security protocols. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 128–142, Doha, Qatar, November 2008. Springer.

[27] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Composition of password-based protocols. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[28] Stéphanie Delaune, Steve Kremer, and Graham Steel. Formal analysis of PKCS#11. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 331–344, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[29] Steve Kremer. Computational soundness of equational theories (tutorial). In Gilles Barthe and Cédric Fournet, editors, *Revised Selected Papers from the 3rd Symposium on Trustworthy Global Computing (TGC'07)*, volume 4912 of *Lecture Notes in Computer Science*, pages 363–382, Sophia-Antipolis, France, 2008. Springer.

[30] Steve Kremer, Antoine Mercier, and Ralf Treinen. Proving group protocols secure against eavesdroppers. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 116–131, Sydney, Australia, August 2008. Springer-Verlag.

[31] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi-calculus. In V. Arvind and Sanjiva Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 133–145, New Delhi, India, December 2007. Springer.

[32] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In Joachim Biskup and Javier Lopez, editors, *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 610–625, Dresden, Germany, September 2007. Springer.

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

[34] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, Venice, Italy, July 2006. IEEE Computer Society Press.

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

[36] Steve Kremer and Mark D. Ryan. Analysing the vulnerability of protocols to produce known-pair and chosen-text attacks. In Riccardo Focardi and Gianluigi Zavattaro, editors, *Proceedings of the 2nd International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'04)*, volume 128 of *Electronic Notes in Theoretical Computer Science*, pages 84–107, London, UK, May 2005. Elsevier Science Publishers.

[37] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Programming Languages and Systems - Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer.

[38] Aybek Mukhamedov, Steve Kremer, and Eike Ritter. Analysis of a multi-party fair exchange protocol and formal proof of correctness in the strand space model. In Andrew S. Patrick and Moti Yung, editors, *Revised Papers from the 9th International Conference on Financial Cryptography and Data Security (FC'05)*, volume 3570 of *Lecture Notes in Computer Science*, pages 255–269, Roseau, The Commonwealth Of Dominica, August 2005. Springer.

[39] Jean Cardinal, Steve Kremer, and Stefan Langerman. Juggling with pattern matching. In Paolo Ferragina and Roberto Grossi, editors, *International Conference on Fun with Algorithms (FUN 2004)*, pages 147–158, Isola d'Elba, Italy, May 2004. Edizioni Plus, Università di Pisa.

[40] Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party contract signing. In Riccardo Focardi, editor, *17th IEEE Computer Security Foundations Workshop*, pages 266–279, Asilomar, CA, USA, June 2004. IEEE Computer Society Press.

[41] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In Jong In Lim and Dong Hoon Lee, editors, *6th International Conference on Information Security and Cryptology (ICISC 2003)*, volume 2971 of *Lecture Notes in Computer Science*, Seoul, Korea, November 2003. Springer-Verlag.

[42] Steve Kremer and Jean-François Raskin. Game analysis of abuse-free contract signing. In Steve A. Schneider, editor, *15th IEEE Computer Security Foundations Workshop*, pages 206–220, Cape Breton, Nova Scotia, Canada, June 2002. IEEE Computer Society Press.

[43] Olivier Markowitch, Dieter Gollmann, and Steve Kremer. On fairness in exchange protocols. In Pil Joong Lee and Chae Hoon Lim, editors, *5th International Conference on Information Security and Cryptology (ICISC 2002)*, volume 2587 of *Lecture Notes in Computer Science*, pages 451–464, Seoul, Korea, November 2002. Springer-Verlag.

[44] Steve Kremer and Olivier Markowitch. Selective receipt in certified e-mail. In C. Pandu Rangan and C. Ding, editors, *Progress in Cryptology - Indocrypt 2001*, volume 2247 of

Lecture Notes in Computer Science, pages 136–148, Chennai, India, December 2001. Springer-Verlag.

[45] Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In Kim G. Larsen and Mogens Nielsen, editors, *Concurrency Theory - CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, August 2001. Springer-Verlag.

[46] Olivier Markowitch and Steve Kremer. An optimistic non-repudiation protocol with transparent trusted third party. In George I. Davida and Yair Frankel, editors, *Information Security Conference 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, October 2001. Springer-Verlag.

[47] Olivier Markowitch and Steve Kremer. A multi-party optimistic non-repudiation protocol. In Dongho Won, editor, *3rd International Conference on Information Security and Cryptology (ICISC 2000)*, volume 2015 of *Lecture Notes in Computer Science*, pages 109–122, Seoul, Korea, December 2000. Springer-Verlag.

Other publications

[48] Stéphanie Delaune, Steve Kremer, Mark D. Ryan, and Graham Steel. A formal analysis of authentication in the TPM (short paper). In Véronique Cortier and Kostas Chatzikokolakis, editors, *Preliminary Proceedings of the 8th International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'10)*, Paris, France, August 2010.

[49] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. In Hubert Comon-Lundh and Catherine Meadows, editors, *Preliminary Proceedings of the 4th International Workshop on Security and Rewriting Techniques (SecReT'09)*, pages 47–58, Port Jefferson, NY, USA, July 2009.

[50] Steve Kremer, Antoine Mercier, and Ralf Treinen. Reducing equational theories for the decision of static equivalence (preliminary version). In Hubert Comon-Lundh and Catherine Meadows, editors, *Preliminary Proceedings of the 4th International Workshop on Security and Rewriting Techniques (SecReT'09)*, Port Jefferson, NY, USA, July 2009.

[51] Ben Smyth, Mark D. Ryan, Steve Kremer, and Mounira Kourjieh. Election verifiability in electronic voting protocols (preliminary version). In Olivier Pereira, Jean-Jacques Quisquater, and François-Xavier Standaert, editors, *Proceedings of the 4th Benelux Workshop on Information and System Security (WISSEC'09)*, Louvain-la-Neuve, Belgium, November 2009.

[52] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Symbolic bisimulation for the applied pi calculus. In Daniele Goria and Catuscia Palamidessi, editors, *Proceedings of the 5th International Workshop on Security Issues in Concurrency (SecCo'07)*,

Electronic Notes in Theoretical Computer Science, Lisbon, Portugal, September 2007. Elsevier Science Publishers.

[53] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In Michael Backes and Yassine Lakhnech, editors, *Proceedings of the 3rd Workshop on Formal and Computational Cryptography (FCC'07)*, Venice, Italy, July 2007.

[54] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying properties of electronic voting protocols. In *Proceedings of the IAVoSS Workshop On Trustworthy Elections (WOTE'06)*, pages 45–52, Cambridge, UK, June 2006.

[55] Steve Kremer. Formal verification of cryptographic protocols. Invited tutorial, 7th School on Modelling and Verifying Parallel Processes (MOVEP'06), Bordeaux, France, June 2006. 5 pages.

[56] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Receipt-freeness: Formal definition and fault attacks (extended abstract). In *Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005)*, Milan, Italy, September 2005.

[57] Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party contract signing. In *Workshop on Issues in the Theory of Security - WITS 2004*, April 2004. co-located with ETAPS 2004.

[58] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. Efficient designated verifier signatures. In Ludo Tolhuyzen, editor, *Twenty-fourth Symposium on Information Theory in the Benelux*, pages 187–194, Veldhoven, The Netherlands, May 2003. Werkgemeenschap Informatie- en Communicatietheorie, Enschede.

[59] Steve Kremer and Olivier Markowitch. A multi-party non-repudiation protocol. In J. Eloff and S. Qing, editors, *15th International Conference on Information Security - Sec 2000*, IFIP World Computer Congress, pages 271–280, Beijing, China, August 2000. Kluwer Academic.

[60] Steve Kremer and Olivier Markowitch. Optimistic non-repudiable information exchange. In J. Biemond, editor, *21st Symp. on Information Theory in the Benelux*, pages 139–146, Wassenaar, The Netherlands, May 2000. Werkgemeenschap Informatie- en Communicatietheorie, Enschede.

[61] Steve Kremer and Jean-François Raskin. Formal verification of non-repudiation protocols - a game approach. In *Workshop on Formal Methods and Computer Securiy - FMCS 2000*, July 2000. co-located with CAV 2000.

[62] Steve Kremer and Jean-François Raskin. A game approach to the verification of exchange protocols - application to non-repudiation protocols. In *Workshop on Issues in the Theory of Security - WITS 2000*, July 2000. co-located with ICALP 2000.

List of co-authors

- Myrto Arapinis
- Mathieu Baudet
- Jean Cardinal
- Rohit Chadha
- Ştefan Ciobâcă
- Véronique Cortier
- Stéphanie Delaune
- Dieter Gollmann
- Mounira Kourjieh
- Ralf Küsters
- Stefan Langerman
- Olivier Markowitch
- Laurent Mazaré
- Antoine Mercier
- Aybek Mukhamedov
- Olivier Pereira
- Jean-François Raskin
- Eike Ritter
- Mark D. Ryan
- Shahrokh Saeednia
- Andre Scedrov
- Ben Smyth
- Graham Steel
- Ralf Treinen
- Bogdan Warinschi
- Jianying Zhou