École Doctorale Sciences Pratiques

Mémoire d'Habilitation à Diriger les Recherches

Stéphanie Delaune

Informatique

Verification of security protocols: from confidentiality to privacy

Composition du Jury :

Martín Abadi	examinateur
David Basin	rapporteur
Gérard Berry	président
Bruno Blanchet	examinateur
Hubert Comon-Lundh	examinateur
Claude Kirchner	rapporteur
Ralf Küsters	rapporteur
John Mitchell	examinateur

Laboratoire Spécification et Vérification (LSV) – UMR 8643

Résumé

Pendant de nombreuses années, les applications liées à la sécurisation de l'information étaient essentiellement militaires. Avec le développement des réseaux de communications comme Internet, et l'essor du commerce électronique, le besoin d'assurer la sécurité des échanges a considérablement augmenté. Les communications « sécurisées » sont réalisées par l'utilisation de petits programmes appelés protocoles cryptographiques. Le problème de la vérification des protocoles cryptographiques est un problème difficile pouvant être vu comme un cas particulier de model-checking où l'environnement considéré est un environnement hostile. De nombreux résultats et outils ont été développés pour permettre la vérification de ces protocoles de manière automatique.

Récemment, pour faire face aux enjeux sociétaux et technologiques, de nouvelles applications ont vu le jour, *e.g.* vote électronique, protocoles de routage dans les réseaux mobiles ad hoc, ... Ces applications présentent des caractéristiques spécifiques qui ne sont pas prises en compte par les outils de vérification existant à l'heure actuelle, *e.g.* utilisation de primitives cryptographiques complexes, propriétés de sécurité lié au respect de la vie privée, ... De plus, il s'avère que les protocoles sont souvent analysés sans tenir compte de l'environment dans lequel ils s'exécutent, ce qui est insuffisant dans la pratique. Dans cette thèse, nous utilisons des méthodes formelles pour étudier ces différents aspects portant sur la vérification de protocoles cryptographiques.

Mots-Clefs

Vérification formelle, protocole cryptographique, respect de la vie privée, équivalence observationnelle, composition

Abstract

Security is a very old concern, which until quite recently was mostly of interest for military purposes. The deployment of electronic commerce changes this drastically. The security of exchanges is ensured by cryptographic protocols which are notoriously error prone. The formal verification of cryptographic protocols is a difficult problem that can be seen as a particular model-checking problem in an hostile environment. Many results and tools have been developed to automatically verify cryptographic protocols.

Recently, new type of applications have emerged, in order to face new technological and societal challenges, *e.g.* electronic voting protocols, secure routing protocols for mobile ad hoc networks, ... These applications involve some features that are not taken into account by the existing verification tools, *e.g.* complex cryptographic primitives, privacy-type security properties, ... This prevents us from modelling these protocols in an accurate way. Moreover, protocols are often analysed in isolation and this is well-known to be not sufficient. In this thesis, we use formal methods to study these aspects concerning the verification of cryptographic protocols.

Keywords

Formal verification, cryptographic protocol, privacy, observational equivalence, composition

Merci !

Je tiens à remercier ici les personnes qui, de près ou de loin, ont contribué à la concrétisation de ce travail. Elles sont nombreuses et il sera difficile de ne pas en oublier. Je vais donc essayer de procéder méthodiquement.

En tout premier lieu, je tiens à remercier tous les membres de mon jury pour leurs questions enrichissantes et nombreuses ! Merci à David Basin, Ralf Küsters, et Claude Kirchner pour avoir accepté la lourde tâche de rapporteurs. Merci à Martín Abadi, Bruno Blanchet, et John Mitchell pour avoir participé à ce jury. Gérard Berry m'a fait l'honneur de présider le jury, et je l'en remercie vivement. Enfin, je tiens à remercier Hubert qui a toujours su orienter mes recherches et sans lequel cette thèse d'habilitation n'aurait pas vu le jour aussi vite.

Je souhaite aussi remercier tous les membres du LSV pour leur disponibilité et leur gentillesse. Je souhaite souligner que le LSV est un endroit où il fait bon vivre et travailler. Les manifestations en tout genre contribue à cette bonne ambiance. J'ai une pensée particulière pour Antoine qui n'aura pas eu le temps de planifier mon habilitation. Merci à toutes les personnes qui m'ont aidé dans l'organisation de cette journée. En particulier, un grand merci à Virginie pour résoudre avec brio les problèmes administratifs et pour sa patience, même si cela me coûte parfois quelques cookies.

Un grand merci à mes co-auteurs pour les discussions scientifiques qui ont menées aux travaux de cette thèse. En particulier, merci à Mark et Véronique pour m'avoir accueilli lors de séjours post-doctoraux au cours desquels j'ai beaucoup appris.

Je souhaite également remercier Graham et Steve avec qui j'ai partagé beaucoup de choses. Outre mon bureau que je partage avec Steve depuis plusieurs années déjà, nous avons partagé tous les trois cette aventure qu'est l'habilitation. Cela nous a permis d'organiser le SECSI Colloquium. J'en profite d'ailleurs pour remercier Hubert à l'initiative de ce projet, ainsi que tous les participants qui ont contribué à faire de ces journées un succès.

Je n'oublie pas les étudiants que j'ai eu la chance d'encadrer et dont les résultats sont mentionnés dans cette thèse. Ils m'ont appris autant (ou peut-être même plus) que j'ai pu leur apprendre. Sans eux, cette thèse ne serait pas ce qu'elle est. En particulier, merci à M$^{\text{elle}}$ Arapinis que j'ai eu l'honneur de co-encadrer pendant sa dernière année de thèse passée au LSV, et qui a contribué non seulement aux résultats mentionnés dans ce manuscrit, mais aussi à l'exposé de cette thèse d'habilitation en m'autorisant à réutiliser certains de ses transparents.

Finalement, un merci particulier à Nico pour son soutien et à Julie pour ses encouragements : « Bravo ». Cela m'a aidé à mener à bien ce projet.

Contents

Chapter 1

Introduction

This habilitation thesis reports on a selection of my contributions since my PhD thesis in 2006. To improve readability, my own publications are cited like [1] whereas the other references are cited like [AB03]. Moreover, for the sake of clarity, some results are only described informally and some others are not presented in this manuscript.

1.1 The growing importance of security protocols

Security is a very old concern, which until quite recently was mostly of interest for military purposes, and therefore of rather limited interest to the public at large. The deployment of electronic commerce changed this drastically. Security protocols are widely used today to secure transactions that take place through public channels like the Internet. Typical functionalities are the transfer of a credit card number or the authentication of a user on a system. Because of their increasing ubiquity in many important applications (*e.g.* electronic commerce, electronic voting, ...), a very important research challenge consists in developing methods and verification tools to increase our trust on security protocols, and so on the applications that rely on them. For example, more than 12 billion Euros are spent each year using Internet transactions. Moreover, new types of protocols are still emerging in order to face new technological and societal challenges. As an illustrative purpose, two applications having an important societal impact are described below.

Electronic voting. Electronic voting promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. It can be used for a variety of types of elections, from small committees or on-line communities to full-scale national elections. For these reasons, governments the world over are trialling and adopting electronic voting systems.

However, recent studies have highlighted inadequacies in implemented systems. For instance, the electronic voting machines used in US elections have been fraught with security problems. Researchers [KSRW04] have analysed the source code of the Diebold machines used in 37 US states. This analysis has produced a catalogue of vulnerabilities and possible attacks. More recent work [FHF06] has produced a security study of the Diebold AccuVote-TS voting machine, including both hardware and software. The results show that it is vulnerable to very serious attacks that can be performed at a large scale.

Mobile ad hoc networking. Over the past decade, wireless, mobile communication technologies have matured and been widely adopted. For instance, the number of cellular phones now exceeds by far that of wired phones. The proliferation of portable computing devices (*e.g.* RFID tags) has lead to a range of new computer security problems that are regularly reported by the media [Goo10]. Whereas RFID tagging could allow many advantages related to production, tracking and tracing of people, animals and products, it cannot be at the expense of health, security, or the fundamental rights to privacy and data protection.

The same kind of things happens in vehicular ad hoc networks where applications such as collision warning systems and high speed toll payment are envisaged to improve road safety. Those applications rely on a beacon signal which poses a threat to privacy since it could allow a vehicle to be tracked. It seems currently socially unacceptable that citizens would be tracked and traced wherever they go, all the time. Moreover, coping with mobility and the volatility of wireless communications in such systems is critical.

The Internet is a large common space, accessible to everyone around the world. As in any public space, people should take appropriate precautions to protect themselves against fraudulent people and processes. It is therefore essential to obtain as much confidence as possible in the correctness of the applications that we use.

1.2 Security protocols

Cryptography is the practice and study of hiding information. Much of the theoretical work in cryptography concerns cryptographic primitives, *i.e.* algorithms with basic cryptographic properties. These primitives provide fundamental properties, which are used to develop more complex tools called security protocols, which guarantee one or more high-level security properties.

1.2.1 Cryptographic primitives

Cryptology prior to the modern age was almost synonymous with encryption, the conversion of information from a readable state to nonsense. The sender retains the ability to decrypt the information and therefore prevents unwanted persons from reading it. Since World War II and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread (*e.g.* cash machine, electronic commerce, electronic voting, ...) The development of digital computers and electronics after World War II made possible much more complex ciphers. Typical examples of cryptographic primitives include one-way functions, symmetric and asymmetric encryptions, ...

Symmetric encryption. Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key. This was the only kind of encryption publicly known until June 1976. For instance, the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs which have been designated cryptography standards by the US government. Despite its deprecation as an official standard, DES remains quite popular; it is used across a wide range of applications, from ATM encryption, to e-mail privacy and secure remote access.

A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different

key, and perhaps each ciphertext exchanged as well. The number of required keys increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all straight and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.

Asymmetric encryption. In 1976, W. Diffie and M. Hellman proposed the notion of public key cryptography in which two different but mathematically related keys are used – a public key and a private key [DH76]. A public key system is so constructed that calculation of one key (the 'private key') is computationally infeasible from the other (the 'public key'), even though they are necessarily related. In public key cryptosystems, the public key may be freely distributed, while its associated private key must remain secret. The public key is typically used for encryption, while the private key is used for decryption. W. Diffie and M. Hellman showed that public key cryptography was possible by presenting the Diffie-Hellman key exchange protocol [DH76]. In 1978, R. Rivest, A. Shamir, and L. Adleman invented RSA, another public key system [RSA78].

Hash function. A hash function takes a message of any length as input, and outputs a short, fixed length hash. Hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as checksums to detect accidental data corruption. For good hash functions, an attacker cannot find two messages that produce the same hash. For instance, MD4 is a long-used hash function which is now broken; MD5, a strengthened variant of MD4, is also widely used but broken in practice. The U.S. National Security Agency developed the Secure Hash Algorithm series of MD5-like hash functions: SHA-0, SHA-1, ... Message authentication codes (MACs) are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt.

New emerging applications such as electronic voting or electronic tolling often rely on some less standard cryptographic primitives to achieve their goals, *e.g.* blind signatures, re-encryption, zero-knowledge proofs, ... It is therefore important to develop verification techniques that are flexible enough to deal with a variety of cryptographic primitives.

1.2.2 Protocols

A protocol can be seen as a list of rules that describes executions; these rules specify the emissions and receptions of messages by the actors of the protocols called agents. They are designed to achieve various security goals such as data privacy and data authenticity, even when the communication between parties takes place over channels controlled by an attacker. For instance, SSL is a protocol that is widely used to provide authentication and encryption in order to send sensitive data such as credit card numbers to a vendor. Those protocols use cryptographic primitives as a building block. A wide variety of cryptographic protocols go beyond the traditional goals of data confidentiality, integrity, and authentication and use more complex cryptographic primitives.

As an illustrative purpose, let us consider the following handshake protocol [Bla01]. This is a simplification of the Denning-Sacco key distribution protocol [DS81], omitting certificates

and timestamps. This protocol allows two agents A and B to share a symmetric key k (freshly generated by A) that can be used for some latter transaction. For instance, B can use this key to encrypt the secret s (freshly generated by B). We consider asymmetric encryption (denoted by the symbol aenc), and signature (denoted by the symbol sign):

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(A)), \mathsf{pk}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

First, the initiator of the protocol generates a fresh symmetric key k. He signs it using his private key $\mathsf{sk}(A)$ before encrypting the whole message with the public key of the responder, i.e. $\mathsf{pk}(B)$. When B receives the message, he decrypts it using his private key $\mathsf{sk}(B)$. He obtains a signature allowing him to authenticate the message. Indeed, intuitively, only A is able to build such a signature: he is the only one who knows $\mathsf{sk}(A)$. Then, from the signature $\mathsf{sign}(k, \mathsf{sk}(A))$, the agent B can extract the key k.

The handshake protocol is well-known to be flawed. In the attack scenario, we assume that the public keys $\mathsf{pk}(A)$, $\mathsf{pk}(B)$, and $\mathsf{pk}(C)$ are public, hence available to all agents. Here, we suppose that C is compromised. Finally, we also assume that the names of the agents, namely A, B, and C are public. Assume that the agent A begins a session with the agent C who is compromised (message 1). This first message allows C to learn the key k_{ac}. Moreover, the agent C can use this first message to contact B on behalf of A (message 2). Hence, B will encrypt his secret s_{ab} (that he wants to share with A only) with the key k_{ac} that is known by C (message 3).

$$\begin{array}{lcl} 1.\ A \to C & : & \mathsf{aenc}(\mathsf{sign}(k_{ac}, sk(A)), \mathsf{pk}(C)) \\ 2.\ C(A) \to B & : & \mathsf{aenc}(\mathsf{sign}(k_{ac}, sk(A)), \mathsf{pk}(B)) \\ 3.\ \quad B \to A & : & \mathsf{senc}(s_{ab}, k_{ac}) \end{array}$$

For a long time, it was believed that designing a strong encryption scheme was sufficient to ensure secure message exchanges. Starting from the 1980's, researchers understood that even with perfect encryption schemes, messages exchanges were still not necessarily secure. This fact is illustrated with the attack described above. Indeed, this attack can be mounted without breaking encryption. This leads to the fact that cryptographic protocols have to be verified and proved before they could be trusted. Note that in presence of honest participants who follow the protocol, the protocol works well. There is no failure in this protocol. However, we have to consider the fact that a malicious agent may want to take advantage of this protocol. This means that we need to verify security protocols in an hostile environment, *i.e.*, in presence of malicious agents who do not necessarily follow the instructions specified by the protocols. This is an essential feature which makes protocol verification a difficult task.

1.3 Verification via formal models

The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion. Here, we will assume that primitives work perfectly, meaning that it is not possible to break them. This does not mean however that the protocols that rely on these primitives are secure. There can still remain some logical attacks (see the attack described in Section 1.2.2 on the handshake protocol).

1.3.1 Computational versus symbolic approach

Two distinct approaches have evolved starting with the early 1980's attempt to ground security analysis of protocols on firm, rigorous mathematical foundations. These two approaches are known as the computational approach and the symbolic approach. The central features of the computational approach are detailed bit-level models for system executions, and the hypothesis of a powerful attacker: security is assessed against arbitrary probabilistic polynomial-time machines. It is generally acknowledged that security proofs in this model offer powerful security guarantees. A serious downside of this approach however is that proofs for even small protocols are usually long, difficult, tedious, and highly error prone.

In this habilitation thesis, we concentrate on the symbolic approach that employs an abstract view of the execution where the messages exchanged by parties are symbolic terms. The resulting models are considerably simpler than those of the computational approach, proofs are therefore also simpler, and can benefit from machine support. However, the abstractions render unclear the security guarantees that are obtained at the end. Recently, significant research efforts have attempted to develop paradigms for cryptographic systems analysis that combines the best of both worlds. Computational soundness aims to establish sufficient conditions under which results obtained using symbolic models imply security under computational models. Research on computational soundness was initiated by M. Abadi and Ph. Rogaway [AR00] in 2000. Through their work it became clear that it is possible to employ the tools and methods specific to the symbolic approach to directly obtain computational security guarantees. The crucial implication is that such guarantees can be obtained without making use of the typical computational proofs. Since then, a plethora of papers have tackled this problem (see [CKW10] for a recent survey on this topic).

1.3.2 A variety of symbolic models

Several symbolic models have been proposed for cryptographic protocols. The first one has been described by D. Dolev and A. Yao [DY81] and several models have been proposed since then. A unified model would enable better comparisons between each result but unfortunately such a model does not exist currently. The reason for having several popular symbolic models probably comes from the fact that symbolic models have to achieve two antagonistic goals. On the one hand, models have to be as fine grained and expressive as possible to capture a large range of applications. One the other hand, models have to remain relatively simple in order to allow the design of verification procedures.

Messages are a key concept in this modelling. Whereas messages are bit-strings in the real-world and in the computational approach, here messages are first-order terms. Constants can be nonces, keys, or agents identities. Functions are concatenation, asymmetric and symmetric encryptions or digital signature. During the protocol execution, the agents exchange messages and the attacker may interfere with these exchanges in order to gain some information, authenticate as another agent, or perform any other type of attack. The attacker is defined by a deduction relation which represents the computation that the attacker can do. In other words, the attacker can only do what is allowed by the specification.

Usually, the perfect cryptography assumption is commonly used to simplify these proofs. When making that hypothesis, the encryption schemes are supposed to be perfect. Hence, it is impossible to deduce a plaintext from its ciphertext without knowing the decryption key. The only deduction rule that an attacker can use to deduce some information from a

ciphertext can be described as follows:

$$\frac{\mathsf{aenc}(x, \mathsf{pk}(y)) \qquad \mathsf{sk}(y)}{x}$$

This rule expresses the fact that an attacker who knows $\mathsf{aenc}(m, \mathsf{pk}(a))$, *i.e.* the encryption of the message m with the public key $\mathsf{pk}(a)$, can decrypt this ciphertext to get m only if he knows the associated private key, namely $\mathsf{sk}(a)$. If the encryption scheme has some other properties, *e.g.* commutativity of the RSA encryption scheme, it is important to specify this properties to avoid missing attacks. This is often done by considering an equational theory. For instance, the equation described below expresses a commutativity property.

$$\mathsf{aenc}(\mathsf{aenc}(x, \mathsf{pk}(y)), \mathsf{pk}(z)) = \mathsf{aenc}(\mathsf{aenc}(x, \mathsf{pk}(z)), \mathsf{pk}(y))$$

Therefore, it is important to develop verification techniques that are flexible enough to deal with a variety of cryptographic primitives and their equational theories in order to cope with the new emerging applications.

Without aiming at an exhaustive list we mention below several symbolic models.

Constraint solving. This technique has been introduced by J. Millen and V. Shmatikov [MS01]. It has been reused by H. Comon-Lundh and others [CLS03, CLCZ10]. This technique is well-adapted to deal with a bounded number of sessions. The principle is quite simple and will be reviewed in details in Chapter 2. We will see that it is flexible enough to deal with a variety of equational theories and also to reflect different constructions that are sometimes needed to model some applications (*e.g.* routing protocols).

Horn clauses. The main goal of this approach is to prove security properties of protocols in an automatic way without bounding the number of sessions or the message space of the protocol. In general, this representation of protocols is approximate in that the application of Horn clauses can be repeated any number of times, while the real protocol repeats each step only once per session. Nevertheless the method was successfully used by many researchers [Wei99, Bla01, CLC03, VSS05] The main advantages of this approach is that we can reuse resolution techniques. This is the underlying model of the ProVerif tool [Bla01] that we have used to perform several case studies.

Multiset rewriting. Several models based on rewriting have been proposed, *e.g.* the result by M. Rusinowitch and M. Turuani [RT03]. The multiset rewriting model has been introduced by J. Mitchell *et al.* [CDL$^+$99, DLM04]. We will use a model based on rewriting in Chapter 3 to analyse security APIs.

Process algebra. Process algebra is a well-known formal framework and actually denotes a family of calculi which have been proposed for describing features of distributed and concurrent systems. This generally leads to a model closer to the implementation. In fact, when applied to security protocol analysis, most of them (*e.g.* spi-calculus [AG97], CSP [Sch96]) rely only on a well-identified subset of primitives. This is not the case of the applied pi calculus [AF01] that has been designed to be flexible in this respect. This calculus is flexible

enough to deal with a variety of equational theories. Several results presented in this thesis have been done in this model. We will present this calculus in Chapter 4.

In these models, a notion of observational equivalence is defined. This notion can be used to express equivalence-based properties such as privacy-type properties.

1.3.3 A variety of security properties

Cryptographic protocols aim at ensuring various security goals, depending on the application. One of the main difficulty when analysing a complex protocol such as an electronic voting protocol is that there are a wide variety of requirements to achieve security. For example, in case of electronic voting, it is important to ensure vote-privacy, *i.e.* the fact that a particular voter voted in a particular way is not revealed to anyone. We have also to ensure that only the voters that are authorized to vote can do that. Moreover, there are other needs which seem to be contradictory: the voters have to be able to test that their votes have been correctly taken into account but it should not be possible for them to prove to a third person the value of their ballot (this is required to avoid ballot selling).

Hence, when we are faced to a new application, the first step is to formally define the security properties that this application is supposed to achieve. This is not an obvious step. Indeed, in general, the security properties mentioned in the specification are expressed in natural language and therefore insufficiently precise.

We can distinguish two main classes of security properties.

Trace-based security properties.

The two most classical security properties are secrecy and authentication. Both are trace-based security properties (or reachability properties) saying that a bad state can not be reached. Most of the verification techniques have been developed for the analysis of these two properties.

Secrecy. This property concerns a message used by the protocol. This is typically a nonce or a secret key that should not become public. Even for this quite simple security property, several definitions have been proposed in the literature. Indeed, a message can be public if the attacker is able to deduce its value (weak secrecy), or it can be public as soon as the attacker is able to distinguish this message from a randomly generated message (strong secrecy). In this latter case, secrecy can be expressed by means of an equivalence (see below).

Authentication. Many security protocols have the aim of authenticating one agent to another: one agent should become sure of the identity of the other. They are also several variants of authentication. A taxomony of these has been proposed by Lowe in [Low97].

However, up-to-date applications need to satisfy some other requirements that have not been well-studied until now. For instance route validity is a crucial property that has to be satisfied by a routing protocol. This can also be expressed as a reachability property. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require the notion of observational equivalence.

Equivalence-based security properties.

Intuitively, two processes P and Q are observationally equivalent if it is not possible for an observer to distinguish them. Observational equivalence is crucial when specifying properties like anonymity, or privacy related properties involved in electronic voting (see Chapter 4). More generally, it is a notion that allows one to express flexible notions of security by requiring observational equivalence between a protocol and an idealized version of it, that magically realizes the desired properties.

Privacy. Frequently, communication between two principals reveals their identities and presence to third parties. Indeed, privacy is in general not one of the explicit goals of common authentication protocol. However, we may want a protocol that achieves this goal. This should mean, in particular, that an attacker cannot distinguish a run between A and B from a run between two other principals A' and B', under appropriate hypotheses. In the context of electronic voting, privacy means that the vote of a particular voter is not revealed to anyone. This is one of the fundamental security properties that an electronic voting system has to satisfy. The same concept is also a desirable property in the context of on-line auction systems. In this situation, privacy means that a third party should not be able to determine the bidding price of any bidder (except for the winning bidder if the winning bid is published).

Unlinkability. Protocols that keep the identity of their users secure may still allow an attacker to identify particular sessions as having involved the same principal. Such linkability attacks may, for instance, make it possible for a third party to trace the movements of someone carrying an RFID tag. Intuitively, protocols are said to provide unlinkability (or untraceability), if an attacker is not able to distinguish a scenario in which the same tag is involved in many sessions from one that involved different tags. Although untraceability is mainly mentioned in the context of RFID systems, linkability attacks are not restricted to this application.

Receipt-freeness. In the context of electronic voting, this property stipulates that 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 [BT94]. Receipt-freeness is intuitively 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.

This property is also required in electronic auction protocols: a bidder should not be able to prove to a third party that he has bid in a certain way. This is required to protect bidders from being coerced to show how they bid.

1.4 Existing results

Many results and tools have been developed to automatically verify cryptographic protocols. However, they mostly concern trace-based security properties and especially (weak) secrecy.

1.4.1 Decidability and undecidability results

Many decidability an undecidability results have been obtained under the perfect cryptography assumption. Recently, several works tried to extend these results to protocols with some algebraic properties. This was the main topic of my PhD thesis [41] in which several decision procedures for a variety of algebraic properties are proposed. However, only trace-based properties are considered. We give here an overview of the existing results (see [9] for more details).

Some undecidability results. Though cryptographic protocols are often described in a concise way, the verification problem is difficult because of many sources of unboundedness in their modelling, for instance the number of sessions, the length of messages, or the nonce generation. As a consequence deciding whether a protocol preserves a secrecy property is undecidable even under the perfect cryptography assumption [EG83, CLC04b, DLMS99, AC02].

Bounded number of sessions. A prominent source of undecidability is the unbounded number of sessions. Actually, in [RT03], M. Rusinowitch and M. Turuani extend the work of R. Amadio *et al.* [ALV02] by giving a co-NP-complete procedure for deciding protocol security for the Dolev-Yao attacker as long as the number of sessions is bounded. Some similar results [MS01, Bor01, CLCZ10] have been obtained in other models. Note that even if it is assumed that there is a bounded number of sessions (thus, also a bounded number of nonces), it is still not easy to design a decision algorithm since the number of messages that can be created by the attacker is unbounded.

Certain algebraic properties of encryption, such as the homomorphic properties of RSA or the properties induced by chaining methods for block ciphers, are widely used in protocol constructions. Many real attacks rely on these properties. Recently, several procedures have been proposed to decide insecurity of cryptographic protocols when considering some algebraic properties of the cryptographic primitives, mostly for a finite number of sessions. For instance, several procedures that deal with protocols relying on the exclusive-or operator have been proposed [CLS03, CKRT03b]. Some properties of the modular exponentiation operator are also taken into account in some decidability results [CKRT03a, MS05].

However, there are very few decision procedures dealing with equivalence-based properties. H. Hüttel [Hut02] shows such a result in a context of process algebra for the notion of strong secrecy expressed as an observational equivalence property. M. Baudet provides also an interesting result showing that an equivalence property (stronger than the standard notion) is decidable for the class of subterm convergent equational theories [Bau05, Bau07], generalizing the result of V. Cortier and M. Abadi obtained in the passive case [AC06]. We will come back to these results in Chapter 5 and Chapter 6 after a presentation of my own contributions in this area.

Unbounded number of sessions. In this setting, to get decidability results, some other restrictions are considered. One of the first results is a PTIME complexity result which has been obtained by D. Dolev *et al.* for ping-pong protocols between two participants [DY81]: in each step of the protocol, one of the agents applies a sequence of operators to the last received message, and sends it to the other agent. Some decidability results in presence of algebraic properties have also been obtained. For instance, in [CLC03], they consider the exclusive-or operator and they assume a finite number of nonces and suppose that at each

transition an agent may copy at most one unknown component of the received message. Some decidable classes have been proposed by K. Verma *et al.* to deal with exclusive-or and Abelian groups [Ver03, VSS05].

Concerning equivalence-based properties, a procedure has been proposed by B. Blanchet *et al.* [BAF08]. This procedure considers a strong notion of equivalence and has been implemented in the ProVerif tool. It allows one to verify some equivalence-based security properties. However, we will see (Part II of this manuscript) that the underlying notion of equivalence is too strong in several situations.

1.4.2 Verification tools

Some of the decision procedures briefly mentioned above have been implemented. Instead of identifying decidable classes, another approach consists of designing semi-decision procedures that work well in practice. This approach has been followed by several authors. Below, we briefly describe some verification tools, focusing on those that are able to handle algebraic properties.

AVISPA tool suite. The AVISPA tool suite [ABB+05] comprises a common high-level input language and four verification backends: OFMC, SATMC, CL-ATSE, and TA4SP. For bounded verification of protocols (using small bounds equal to two or three runs), AVISPA is state-of-the-art in terms of both speed and features. In particular, some of the backends support some algebraic properties, allowing for correct modelling of exclusive-or, or Diffie-Hellman exponentiation.

ProVerif. ProVerif is an automatic verifier developed by B. Blanchet [Bla01]. Any property can be tested provided that it can be expressed in Horn clauses. The treatment of equations is quite simple. In particular, the system may not terminate if some complex equations are entered. In particular, it is not able to deal with exclusive-or. Nevertheless this tool has been successful to analyse many protocols and it is quite flexible to analyse security properties. In particular, it is possible to express strong secrecy and more generally some equivalence-based properties.

Scyther. This tool can verify security protocols for an unbounded number of sessions in less than a second [Cre08a]. Since no approximation methods are used, all attacks found are real attacks on the model. In case where unbounded verification cannot be determined, the algorithm functions as a classical bounded verification tool, and yields results for a bounded number of sessions. However, the tool is not able to deal with algebraic properties and equivalence-based security properties.

Maude-NRL Protocol Analyser. Maude-NRL Protocol Analyzer (Maude-NPA) is an analysis tool for cryptographic protocols that takes into account many of the algebraic properties of cryptosystems that are not included in other tools [EMM07]. These include cancellation of encryption and decryption, Abelian groups (including exclusive-or), and exponentiation. Maude-NPA uses an approach similar to the original NRL Protocol Analyzer of C. Meadows [Mea96]; it is based on unification, and performs backwards search from a final state to determine whether it is reachable. Unlike the original NPA, it has a theoretical basis

in rewriting logic and narrowing, and offers support for a wider basis of equational theories that includes associative-commutative (AC) theories.

1.5 Contributions

Recently, new types of protocols have emerged in order to face new technological and societal challenges, *e.g.* electronic voting protocols and secure routing protocols in wireless ad hoc networks. These applications involve some features that are not taken into account by the existing verification tools preventing us to model these protocols in an accurate way, *e.g.* complex cryptographic primitives, equivalence-based security properties, ... Moreover, we will see that, in the symbolic approach, protocols are often analysed in isolation and this is well-known to be not sufficient to obtain guarantee in an environment where some other protocols (possibly sharing some long-term keys) are executed as well.

1.5.1 Content of the thesis

The thesis is organized in three parts. The content of each part is briefly described below.

Part I. This first part is devoted to the study of trace-based properties. In **Chapter** 2, we consider the constraint solving approach and we demonstrate the flexibility of this approach by extending it in two main directions. This allows us to deal with trace-based properties (secrecy, route validity, ...) for a bounded number of sessions. First, we extend deducibility constraints with membership constraints in order to deal with the blind signature equational theory, an equational theory that we have encountered during our study of electronic voting protocols. Then, we consider another application area, namely routing protocols, and we show again how to extend this approach to deal with neighbourhood constraints. In **Chapter** 3, we consider an unbounded number of sessions focusing on the verification of security APIs.

Part II. The second part of this habilitation thesis is devoted to the study of privacy-type properties. First, in **Chapter** 4, we introduce the applied pi calculus. This is a process algebra flexible enough to model a variety of cryptographic primitives. This allows us in particular to model electronic voting protocols that often rely on complex cryptographic primitives. We show how to model privacy type properties in this setting.

In **Chapter** 5, we propose several decision procedures together with their implementations for deciding indistinguishability of sequences of messages. We also present a combination result. Altogether, this allows us to deal with a variety of equational theories.

The results presented in the previous chapter do not allow one to take into account the dynamic behaviour of the protocol. In **Chapter** 6, we consider an active attacker who may interact with the protocol and we propose several techniques and decision procedures to decide privacy-type properties in this setting.

Part III. In a last part, we investigate composition results. More precisely, we focus on parallel composition under shared secrets. We consider composition of different protocols and also composition of different sessions coming from the same protocols. In both cases, we propose a simple transformation that makes it possible to transform a protocol that is safe in

isolation into a protocol that can safely be used in an environment where some other protocols are executed as well. This allows us to prove a protocol in isolation or even for one session (we have seen that this is already a challenging problem) and to obtain guarantee even if the protocol is executed in a more general setting. We focus on trace-based properties and we consider also the particular case of password-based protocols. We provide a way to ensure security of the protocols even if a user chooses the same password for different applications.

1.5.2 Collaborations

The results presented in this habilitation thesis have been obtained in collaboration with many other researchers that are listed below:

Myrto Arapinis	Véronique Cortier
Mathilde Arnaud	Morten Dahl
Mathieu Baudet	Jérémie Delaitre
Sergiu Bursuc	Steve Kremer
Rohit Chadha	Olivier Pereira
Vincent Cheval	Mark D. Ryan
Ştefan Ciobâcă	Ben Smyth
Hubert Comon-Lundh	Graham Steel

In particular, this habilitation thesis describes the results obtained during the supervision of three PhD thesis:

- Sergiu Bursuc defended his PhD thesis in December 2009. His thesis was co-supervised with H. Comon-Lundh and was about "Verification of security protocols and equational theories". Some of the results are described in Chapter 2 (Section 2.2). The other results obtained in collaboration with S. Bursuc and H. Comon-Lundh have not been reported in this habilitation thesis [35, 34].

- Mathilde Arnaud is currently finishing her PhD thesis. She started in October 2008. Her thesis is co-supervised with V. Cortier and is about "Verification of routing protocols in mobile ad hoc networks". The results obtained so far are described in Chapter 2 (Section 2.3) and have been published in [16]. We have also another joint publication [32] whose main results are described in Chapter 5 (Section 5.4). Actually, Mathilde Arnaud did an internship at LORIA (supervised by V. Cortier) and she obtained some results allowing one to combine some specific equational theories (August 2006). I have generalized these results to a broader class of equational theories when I was a post-doc at LORIA (January 2007).

- Vincent Cheval began his PhD thesis in September 2009. I co-supervise his thesis about "Verification of equivalence-based security properties" with H. Comon-Lundh. The results we have obtained on this topic are described in Chapter 6 (Section 6.3) and have been published in [17].

The works presented in this habilitation thesis are also based on results obtained during the supervision of some internships and master theses. The works presented in Chapter 7 (Section 7.1) have been obtained during the master thesis of Jérémie Delaitre (co-supervised with V. Cortier) and has been published in [29, 7]. I also co-supervised with S. Kremer, the

master thesis of Ştefan Ciobâcă. This work is described in Chapter 5 (Section 5.2) and has been published in [20, 3]. In 2008, Myrto Arapinis obtained a teaching position (ATER) for one year in the computer science department of my laboratory. During this year, she finished her PhD thesis. Actually, I co-supervised her (with Steve Kremer). The result we obtained together is part of her thesis. This result is described in Chapter 7 (Section 7.2) and has been published in [24]. In 2009, Morten Dahl, PhD student at the Aalborg university (Danemark) under the supervision of Hans Hüttel, visited LSV during one year and worked on "Verification of security protocols in Vehicular Ad-Hoc Networks" (with co-advisor Graham Steel). This work is described in Chapter 4 (Section 4.3) and Chapter 6 (Section 6.1).

Part I

Reachability security properties

Chapter 2

Verification via constraint solving

Contents

Constraint systems are quite common (see *e.g.* [CLS03, CZ06]) in modelling security protocols. A constraint system represents the possible executions of a protocol once an interleaving has been fixed. They are used, for instance, to specify secrecy preservation of security protocols under a particular, finite scenario. In case of a bounded number of sessions, M. Rusinowitch and M. Turuani have established a small attack property and they have shown that the problem of deciding secrecy preservation for a fixed set of primitives is co-NP-complete [RT03]. Independently, J. Millen and V. Shmatikov [MS01] designed a constraint solving procedure leading to another algorithm. Since then, the concept of constraint systems has been reused in many works (for instance [CLS03, CLCZ10, MS05] and [8]) to obtain decidability results for different kind of primitives, *e.g.* exclusive-or, Abelian group operator, Diffie-Hellman exponentiation, monoidal equational theories, ...

First, we recall some basic notions before we present a refinement of a decision procedure first proposed by H. Comon-Lundh *et al.* [CLCZ10]. Then, we present two extensions allowing one to model some other classes of protocols. The first one, described in Section 2.2, allows us to consider more primitives. In particular, we deal with the primitive of blind signatures

$$(\mathsf{P})\ \frac{x \quad y}{\langle x, y\rangle} \qquad (\mathsf{SE})\ \frac{x \quad y}{\mathsf{senc}(x, y)}$$

$$(\mathsf{Proj}_1)\ \frac{\langle x_1, x_2\rangle}{x_1} \qquad (\mathsf{Proj}_2)\ \frac{\langle x_1, x_2\rangle}{x_2} \qquad (\mathsf{SD})\ \frac{\mathsf{senc}(x, y) \quad y}{x}$$

Figure 2.1 - intruder deduction system for symmetric encryption and pairing.

used in electronic voting protocols. The goal of the second extension is to allow more constructions in our process algebra in order to model ad hoc routing protocols (see Section 2.3). In both cases, we extend the definition of constraint systems and we revisit the constraint simplification rules to cope with these new constructions.

2.1 The setting

2.1.1 Messages and attacker capabilities

Cryptographic primitives are represented by *function symbols* together with their *arity*, denoted by $ar(\mathsf{f})$. For instance, we may want to consider the function symbols senc (for symmetric encryption) and $\langle \_, \_ \rangle$ for pairing, both with arity 2. We fix an infinite set of variables $\mathcal{X} = \{x, y, \ldots\}$ and an infinite set of names $\mathcal{N} = \{a, b, n, \ldots\}$ that typically represent nonces or agent names. *Terms* are obtained by repeated applications of function symbols on names and variables. For instance, the term $\mathsf{senc}(n, k)$ represents the nonce n encrypted with the key k.

As usual, we write $var(t)$ for the set of variables occurring in t. A term is *ground* if it has no variables. We write $\mathsf{St}(t)$ for the set of *subterms* of a term t. *Substitutions* are written $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $dom(\sigma) = \{x_1, \ldots, x_n\}$. The substitution σ is *ground* if each t_i is a ground term. The application of a substitution σ to a term t is written $t\sigma$. If t_1 and t_2 are terms then a *unifier* of t_1 and t_2 is a substitution σ such that $t_1\sigma = t_2\sigma$. It is well-known that unifiable terms have a *most general unifier* (mgu), *i.e.* a substitution σ such that $\sigma \leq \tau$ (there exists a substitution ρ such that $\sigma\rho = \tau$) for any other unifier τ of t_1 and t_2.

The ability of the attacker is modelled by a deduction system. For instance, a deduction system representing the usual Dolev-Yao [DY81] rules for symmetric encryption and pairing is given in Figure 2.1. The first line describes the *composition* rules. We call these deduction rules pairing (P), and symmetric encryption (SE) respectively. The second line describes the *decomposition* rules. These rules are called first and second projections, and symmetric decryption respectively. Intuitively, these deduction rules say that an attacker can compose messages by pairing and encrypting messages provided he knows the encryption key. Conversely, he can decompose messages by projecting or decrypting provided he knows the decryption key.

A term u is *deducible* from a set of terms T, denoted by $T \vdash u$, if there exists a *proof* π, *i.e.* a tree such that the root of π is labelled with u, the leaves of π are labelled with $v \in T$ and every intermediate node is an instance of one of the rules of the deduction system. For

instance, the term $\langle k_1, k_2 \rangle$ is deducible from the set $T_1 = \{\mathsf{senc}(k_1, k_2), k_2\}$ using the inference system given in Figure 2.1.

We let $\mathtt{step}(\pi)$ be the set of terms labelling the proof π and $\mathtt{leaves}(\pi)$ be the multiset of the terms that label the leaves of π. If π is a proof, we let $\mathtt{last}(\pi)$ be the last inference step in π, $\mathtt{premises}(\pi)$ be the proofs of the premises of $\mathtt{last}(\pi)$ and $\mathtt{conc}(\pi)$ be its conclusion. More formally, we have that:

$$\text{if} \quad \pi = \frac{\pi_1 \quad \cdots \quad \pi_n}{u} \quad \text{then} \quad \begin{cases} \mathtt{last}(\pi) = \dfrac{\mathtt{conc}(\pi_1) \quad \cdots \mathtt{conc}(\pi_n)}{u} \\ \mathtt{premises}(\pi) = \{\pi_1, \ldots, \pi_n\} \\ \mathtt{conc}(\pi) = u \end{cases}$$

2.1.2 Protocols insecurity as a constraint solving problem

We show on an example how protocol insecurity is reduced to constraint solving. A cryptographic protocol is defined by a set of programs (or *roles*) which may be executed by agents distributed over a network. In the simplest case these programs are linear sequences of *receive* and *send* instructions on a public communication channel. The attacker may modify the messages sent on the channel using a certain set of *attacker capabilities*, *e.g.* the rules described in Figure 2.1. The fact that all messages may be modified by the attacker is often expressed by saying that *the attacker is the network*. The most basic property of cryptographic protocols is the so-called *secrecy* property, which states that for any number of agents executing the roles, for any possible interleaving of the program execution, and for any modifications of the messages by the attacker (according to his deduction capabilities) the attacker is not able to deduce a certain message which is supposed to remain secret.

In the case of a bounded number of sessions, *i.e.* a bounded number of role instances running in parallel, there is only a bounded number of symbolic traces, each of which represents an interleaving of the execution of the parallel role instances. Every message received during the execution of a role is a message that can be deduced using the intruder deduction capabilities from the messages sent before on the communication channel. The idea of the verification algorithm is to guess a symbolic trace in which the messages are represented by terms containing variables. This symbolic trace corresponds to a concrete execution trace if the variables can be instantiated in such a way that at every moment a message received by an agent can in fact be deduced by the attacker from the messages seen before.

Let us consider the handshake protocol introduced in Chapter 1, temporarily considering asymmetric encryption and signature:

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(A)), \mathsf{pk}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

This protocol is used by roles A and B to share a secret s (freshly generated by B) that can be used for some later transaction. The protocol involves a symmetric key k freshly generated by A, the signature key of A, denoted $\mathsf{sk}(A)$, and the public key of B, denoted $\mathsf{pk}(B)$. The fact that the execution of a single session of the protocol is insecure is described by the following

sequence of deduction constraints:

$$\mathcal{C}_{\mathsf{handshake}} \stackrel{\mathsf{def}}{=} \begin{cases} T_{\mathsf{init}},\ \mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),\mathsf{pk}(c)) \stackrel{?}{\vdash} \mathsf{aenc}(\mathsf{sign}(x,\mathsf{sk}(a)),\mathsf{pk}(b)) \\ T_{\mathsf{init}},\ \mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),\mathsf{pk}(c)),\ \mathsf{senc}(s,x) \stackrel{?}{\vdash} \mathsf{senc}(y,k) \\ T_{\mathsf{init}},\ \mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),\mathsf{pk}(c)),\ \mathsf{senc}(s,x) \stackrel{?}{\vdash} s \end{cases}$$

where T_{init} is the initial knowledge of the attacker. The last deduction step states that the secret is revealed and the protocol is insecure if the constraint system has a solution. In this scenario, we consider one instance of the role A (with agent parameters a and c), and one instance of the role B (with agent parameters b and a). In this scenario, we assume that the keys $\mathsf{pk}(a)$, $\mathsf{pk}(b)$, and $\mathsf{pk}(c)$ are public, hence available to all agents, including the attacker. Here, we suppose that c is compromised: this means that his private key $\mathsf{sk}(c)$ is also available to the attacker. Finally, the attacker knows also the names of the agents, namely a, b, and c. Thus, the total initial attacker knowledge is:

$$T_{\mathsf{init}} = \{\mathsf{pk}(a),\ \mathsf{pk}(b),\ \mathsf{pk}(c),\ \mathsf{sk}(c),\ a,\ b,\ c\}.$$

Definition 2.1 (constraint system) *A* constraint system $\mathcal{C}$ *is either* $\bot$ *or a finite sequence of expressions* $T_1 \stackrel{?}{\vdash} u_1, \ldots, T_n \stackrel{?}{\vdash} u_n$*, called* constraints*, where each* T_i *is a non empty set of terms, called the* left-hand side *of the constraint and each* u_i *is a term, called the* right-hand side *of the constraint, such that:*

- monotonicity*:* $T_i \subseteq T_{i+1}$ *for every* i *such that* $1 \leq i < n$*;*
- origination*: if* $x \in var(T_i)$ *for some* i *then there exists* $j < i$ *such that* $x \in var(u_j)$*.*

A solution *of* $\mathcal{C}$ *is a ground substitution* θ *such that for every* $(T \stackrel{?}{\vdash} u) \in \mathcal{C}$*, we have that* $T\theta \vdash u\theta$*. The empty constraint system is always satisfiable whereas* $\bot$ *denotes an unsatisfiable system.*

A constraint system $\mathcal{C}$ is usually denoted as a conjunction of constraints:

$$\mathcal{C} = T_1 \stackrel{?}{\vdash} u_1 \ \wedge\ \ldots\ \wedge\ T_n \stackrel{?}{\vdash} u_n$$

with $T_i \subseteq T_{i+1}$, for all $1 \leq i < n$. The second condition in Definition 2.1 says that each time a variable occurs in some left-hand side, it must have occurred before in some right-hand side. The left-hand side of a constraint system usually represents the messages sent on the network.

Coming back to our example, we have that $\mathcal{C}_{\mathsf{handshake}}$ is a constraint system. Moreover, the substitution $\sigma = \{x \mapsto k,\ y \mapsto s\}$ is a solution of $\mathcal{C}_{\mathsf{handshake}}$. Indeed, considering an inference system modelling also asymmetric encryption and signature, we have that $\mathsf{sign}(k,\mathsf{sk}(a))$ and k are deducible from $\mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),\mathsf{pk}(c))$. Then the attacker can compute $\mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),\mathsf{pk}(b))$. It is easy to see that the two last constraints are also satisfied.

Hence, we get the following problem, whose decision is the subject of this chapter:

> Given an inference system and a constraint system $\mathcal{C}$, does there exist a substitution σ such that σ is a solution of $\mathcal{C}$? When it is possible, we also want to find an effective representation of all solutions.

R_1 :	$\mathcal{C} \wedge T \overset{?}{\vdash} u$	$\rightsquigarrow$	$\mathcal{C}$	if $T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\} \vdash u$
R_2 :	$\mathcal{C} \wedge T \overset{?}{\vdash} u$	$\rightsquigarrow_\sigma$	$\mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$	if $\sigma = \mathsf{mgu}(t,u)$ where $t \in \mathsf{St}(T)$, $t \neq u$, and t, u are neither variables nor pairs
R_3 :	$\mathcal{C} \wedge T \overset{?}{\vdash} u$	$\rightsquigarrow_\sigma$	$\mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$	if $\sigma = \mathsf{mgu}(t_1,t_2)$, $t_1, t_2 \in \mathsf{St}(T)$, $t_1 \neq t_2$, and t_1, t_2 are neither variables nor pairs
R_4 :	$\mathcal{C} \wedge T \overset{?}{\vdash} u$	$\rightsquigarrow$	$\bot$	if $var(T \cup \{u\}) = \emptyset$ and $T \not\vdash u$
R_5 :	$\mathcal{C} \wedge T \overset{?}{\vdash} \mathsf{f}(u,v)$	$\rightsquigarrow$	$\mathcal{C} \wedge T \overset{?}{\vdash} u \wedge T \overset{?}{\vdash} v$	for $\mathsf{f} \in \{\langle\rangle, \mathsf{senc}\}$

Figure 2.2 - Simplification rules for symmetric encryption and pairing

2.1.3 Refinement of an existing decision procedure

In [29, 7], we have refined an existing decision procedure for solving constraint systems. Several decision procedures already exist [MS01, CLS03, CZ06, RT03] for solving constraint systems. Some of them [MS01, CLS03, CZ06] are based on a set of simplification rules allowing a general constraint system to be reduced to some simpler one, called *solved*, on which satisfiability can be easily decided. A constraint system is said *solved* [CZ06] if it is different from $\bot$ and if each of its constraints is of the form $T \overset{?}{\vdash} x$, where x is a variable. Note that the empty constraint system is solved. Solved constraint systems are particularly simple since they always have a solution. Indeed, let T_1 be the smallest (*w.r.t.* inclusion) left-hand side of a constraint. From the definition of a constraint system we have that T_1 is non empty and has no variable. Let $t \in T_1$. Then the substitution τ defined by $x\tau = t$ for every variable x is a solution since $T \vdash x\theta$ for any constraint $T \overset{?}{\vdash} x$ of the solved constraint system.

The *simplification rules* we consider are described in Figure 2.2. We consider the inference system described in Figure 2.1. In [29, 7], we give a set of simplification rules allowing us to deal with more primitives (asymmetric encryption, signature, ...). All the rules are indexed by a substitution (when there is no index then the identity substitution is implicitly considered). We write $\mathcal{C}_0 \rightsquigarrow_\sigma^* \mathcal{C}_n$ if there are $\mathcal{C}_1, \ldots, \mathcal{C}_{n-1}$ such that $\mathcal{C}_0 \rightsquigarrow_{\sigma_0} \mathcal{C}_1 \rightsquigarrow_{\sigma_1} \ldots \mathcal{C}_{n-1} \rightsquigarrow_{\sigma_n} \mathcal{C}_n$ and $\sigma = \sigma_0\sigma_1 \ldots \sigma_n$. Our rules are the same as in [CZ06] except that we forbid unification of terms headed by $\langle\ \rangle$. Correction and termination are still ensured by [CZ06] and we show that they still form a complete decision procedure. Intuitively, unification between pairs is useless since pairs can be decomposed in order to perform unification on their components. Then, it is possible to build again the pair if necessary. Note that this is not always possible for encryption since the key used to decrypt or encrypt may be unknown by the attacker. Proving that forbidding unification between pairs still leads to a complete decision procedure required in particular to introduce a new notion of minimality for tree proofs for deduction. This procedure has been designed to establish some composition results that are presented in Chapter 7. However, this result is of independent interest. Indeed, we provide a more efficient decision procedure for solving constraint systems, thus for deciding secrecy for a

bounded number of sessions. Of course, the theoretical complexity remains the same (NP).

Theorem 2.1 *[29, 7] Let $\mathcal{C}$ be an unsolved constraint system.*

1. *(Soundness) If $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ for some constraint system $\mathcal{C}'$ and some substitution σ and if θ is a solution of $\mathcal{C}'$ then $\sigma\theta$ is a solution of $\mathcal{C}$.*

2. *(Completeness) If θ is a solution of $\mathcal{C}$, then there exist a solved constraint system $\mathcal{C}'$ and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ and θ' is a solution of $\mathcal{C}'$.*

3. *(Termination) There is no infinite chain $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}_n \rightsquigarrow_{\sigma_{n+1}} \ldots$.*

Hence, given a constraint system $\mathcal{C}_0$, we are able to compute a finite set of solved constraint systems $\mathcal{C}_1, \ldots, \mathcal{C}_n$ together with their associated substitutions $\sigma_1, \ldots, \sigma_n$ such that:

$$\{\theta \mid \theta \text{ is a solution of } \mathcal{C}_0\} = \bigcup_{i=1}^{n} \{\sigma_i\theta' \mid \theta' \text{ is a solution of } \mathcal{C}_i\}.$$

2.2 Local theories

The inference system presented in Figure 2.1 is *local* (see [McA93]). This means that when u is deducible from T, there exists a proof π witnessing this fact that only contains terms that occur in the subterms of u or T. This is why deciding whether a message can be computed by an attacker from a finite set of messages can be performed in polynomial (actually linear) time in this inference system. The starting point of the work presented in this section is the problem of lifting the results obtained for a ground deducibility constraint to general constraint systems (see Definition 2.1). For security protocols, this corresponds to moving from a passive attacker to an active attacker.

Moreover, we want to prove that, for any inference system that is saturated in some suitable way (we call it *good*), the deducibility constraints are decidable. Actually, we prove more: we provide with a constraint simplification algorithm that yields *solved forms*. This allows us not only to decide the existence of a solution, but also to represent all solutions. Such a feature is used in [CLCZ10] for deciding trace properties such as authentication and key cycles in security protocols, and also in [KK05] for deciding game-theoretic security properties such as abuse-freeness. Our results generalise [CLCZ10] to any good inference system that is finite (see Section 2.2.2). However, when the underlying inference system is not finite, our simplification procedure is infinitely branching. We show how to solve this issue in the case of blind signatures by introducing a new kind of constraint (see Section 2.2.3).

2.2.1 Good inference systems

In the following definitions, we introduce our notion of saturation. Informally, if there is a proof such that some intermediate step is too large (we call this a *bad proof* and the large step is called a *bad pattern*), then there must be a simpler proof of the same statement.

If $\mathsf{R} = \dfrac{s_1 \quad \cdots \quad s_n}{s_0}$ is an inference rule, we let $\mathsf{Max}(\mathsf{R})$ be the multiset of the maximal terms s_i, *w.r.t.* the subterm ordering $\lhd$.

Definition 2.2 (bad proof / pattern) *A* bad proof *is a proof π of the form:*

$$\dfrac{u_1 \;\cdots\; u_n \qquad \dfrac{v_1 \;\cdots\; v_m}{u_{n+1}}\,\mathsf{R}_1 \qquad u_{n+2} \;\cdots\; u_{n+k}}{v}\,\mathsf{R}_2$$

such that $\mathsf{R}_1 = \dfrac{s_1 \;\cdots\; s_m}{s}$, $\mathsf{R}_2 = \dfrac{t_1 \;\cdots\; t_{n+k}}{t}$, $s \in \mathsf{Max}(\mathsf{R}_1)$ *and* $t_{n+1} \in \mathsf{Max}(\mathsf{R}_2)$.
A bad pattern *in a proof π is a subproof of π of the form:*

$$\dfrac{\pi_2^1 \;\cdots\; \pi_2^{i-1} \qquad \dfrac{\pi_1^1 \;\cdots\; \pi_1^m}{\mathtt{conc}(\pi_2^i)}\,\mathsf{R}_1 \qquad \pi_2^{i+1} \;\cdots\; \pi_2^n}{v}\,\mathsf{R}_2$$

such that the following proof is a bad proof.

$$\dfrac{\mathtt{conc}(\pi_2^1) \;\cdots\; \mathtt{conc}(\pi_2^{i-1}) \qquad \dfrac{\mathtt{conc}(\pi_1^1) \;\cdots\; \mathtt{conc}(\pi_1^m)}{\mathtt{conc}(\pi_2^i)}\,\mathsf{R}_1 \qquad \mathtt{conc}(\pi_2^{i+1}) \;\cdots\; \mathtt{conc}(\pi_2^n)}{v}\,\mathsf{R}_2$$

If $\pi = \mathsf{R}\theta$ is an instance of an inference rule R and $\mathsf{Max}(\mathsf{R}) = \{s_1, \ldots, s_k\}$, then $\mu(\pi)$ is the multiset $\{s_1\theta, \ldots, s_k\theta\}$. If π is a proof, $\mu(\pi)$ is defined as the multiset of $\mu(\pi')$ for all inference steps π' of π. Formally, if $\mathtt{premises}(\pi) = \{\pi_1, \ldots, \pi_n\}$, we have that:

$$\mu(\pi) = \mu(\pi_1) \uplus \cdots \uplus \mu(\pi_n) \uplus \mu(\mathtt{last}(\pi)).$$

Multisets are ordered using the multiset extensions of their elements: if $\succeq$ is an ordering, we let $\succeq_m$ be its multiset extension.

Definition 2.3 (good inference system) *An inference system is* good *if there is a total well-founded extension $\prec$ of the subterm ordering $\lhd$ such that, for any bad proof π, there is a proof π' of $\mathtt{leaves}(\pi) \vdash \mathtt{conc}(\pi)$ with $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ (multiset inclusion), $\mu(\pi')\ (\prec_m)_m\ \mu(\pi)$.*

In a good inference system, we can restrict our attention to proofs that do not contain bad patterns. More formally, if there exists a proof π of $T \vdash u$, then there exists one that does not contain any bad pattern.

The Dolev-Yao inference system described in Figure 2.1 is good. Another example in which the good inference system is no longer finite is the case of blind signatures as modelled in [KR05]. The inference system is not good. However, following the work by D. Basin and H. Ganzinger [BG01] in which they proved that locality is equivalent to a saturation property of the set of inference rules, we may complete it and get an infinite good inference system.

Example 2.1 *We consider the following rules:*

$$(\mathsf{S})\quad \frac{x \quad y}{\mathsf{sign}(x,y)} \qquad (\mathsf{C})\quad \frac{\mathsf{sign}(x,y)}{x}$$

$$(\mathsf{B})\quad \frac{x \quad y}{\mathsf{blind}(x,y)} \qquad (\mathsf{UB}_1)\quad \frac{\mathsf{blind}(x,y) \quad y}{x} \qquad (\mathsf{UB}_2)\quad \frac{\mathsf{sign}(\mathsf{blind}(x,y),z) \quad y}{\mathsf{sign}(x,z)}$$

Because of the rule (UB_2)*, the system not good. The following proof* π *is bad:*

$$\dfrac{\dfrac{\mathsf{sign}(\mathsf{blind}(\mathsf{blind}(x,x_1),x_2),y) \quad x_2}{\mathsf{sign}(\mathsf{blind}(x,x_1),y)} \quad x_1}{\mathsf{sign}(x,y)}$$

There is no other proof π' *of* $\mathtt{leaves}(\pi) \vdash \mathsf{sign}(x,y)$ *such that* $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ *and in which the term* $\mathsf{sign}(\mathsf{blind}(x,x_1),y)$ *is not used. Thus, for any total well-founded extension* $\prec$ *of* $\lhd$*, there is no proof* π' *of* $\mathtt{leaves}(\pi) \vdash \mathsf{sign}(x,y)$ *such that* $\mathtt{leaves}(\pi') \subseteq \mathtt{leaves}(\pi)$ *and* $\mu(\pi')\ (\prec_m)_m\ \mu(\pi)$*. However, we may add to the inference system shortcuts to replace the bad proofs. Let* $\mathsf{b}_n(x,x_1,\ldots,x_n)$ *be defined by* $\mathsf{b}_1(x,x_1) = \mathsf{blind}(x,x_1)$ *and* $\mathsf{b}_{n+1}(x,x_1,\ldots,x_{n+1}) = \mathsf{blind}(\mathsf{b}_n(x,x_1,\ldots,x_n),x_{n+1})$*. We add the following rules (for every* $n \geq 1$*) and the resulting system is a good inference system.*

$$(\mathsf{UB}) \quad \dfrac{\mathsf{sign}(\mathsf{b}_n(x,x_1,\ldots,x_n),y) \quad x_1 \quad \ldots \quad x_n}{\mathsf{sign}(x,y)}$$

We now only consider good inference systems. The rules of such systems can be divided in three sets.

- The *composition rules* whose conclusion is the only maximal term. Any rule of the form $\dfrac{x_1 \quad \ldots \quad x_n}{\mathsf{f}(x_1,\ldots,x_n)}$ is a composition, *e.g.* (P), (SE), (S).
- The *decomposition rules* whose all maximal terms are premises, *e.g.* (SD).
- The *versatile rules* whose both the conclusion and some premises are maximal, *e.g.* (UB_2).

In what follows, we also assume that:

1. any composition rule has a conclusion $\mathsf{f}(x_1,\ldots,x_n)$ where $x_1,\ldots,x_n$ are variables: each function symbol is either public (and there is such a rule) or private.
2. any versatile rule satisfies the following properties:
   (a) each strict subterm of the conclusion is a subterm of some premise.
   (b) each premise that is not maximal in the rule is a strict subterm of another premise of that rule.

These conditions are satisfied by the inference system given in Figure 2.1 as well as the one described in Example 2.1. Besides these examples, any intruder theory that can be presented by a finite subterm convergent rewrite system satisfies our hypotheses. These hypotheses might not be necessary for our result, but we use them in our proof.

We now classify the proofs, according to the type of the last proof step. This generalises the classical composition/decomposition classification:

Lemma 2.1 (locality lemma) *[18] Let* π *be a proof of* $T \vdash u$ *without bad pattern, one of the following occurs:*

- $\mathtt{last}(\pi)$ *is an instance of a composition rule and* $\mathtt{step}(\pi) \subseteq \mathsf{St}(T \cup \{u\})$*;*

R_{Ax} $\quad \mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma$ $\quad$ where $\sigma = \mathsf{mgu}(u, v)$, $v \in T$ and $u \notin \mathcal{X}$

$\mathsf{R}_{\mathsf{Triv}}$ $\quad \mathcal{C} \wedge T \overset{?}{\vdash} x \wedge T' \overset{?}{\vdash} x \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} x$ $\quad$ when $T \subseteq T'$

$\mathsf{R}_{\mathsf{Compo}}$ $\quad \mathcal{C} \wedge T \overset{?}{\vdash} \mathsf{f}(u_1, \ldots, u_n) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u_1 \wedge \ldots \wedge T \overset{?}{\vdash} u_n$ if f is a public symbol

$\mathsf{R}_{\mathsf{Decompo}}$ $\quad \mathcal{C} \wedge T \overset{?}{\vdash} v \rightsquigarrow \mathcal{C}\theta \wedge T\theta \overset{?}{\vdash} w_1\theta \wedge \ldots T\theta \overset{?}{\vdash} w_n\theta \wedge T'\theta \overset{?}{\vdash} v_1\theta \wedge \ldots \wedge T'\theta \overset{?}{\vdash} v_m\theta$

where:

- $\mathsf{R} = \dfrac{v_1 \ \ldots \ v_m \ \ w_1 \ \ldots \ w_n}{w}$ is a fresh renaming of a decomposition or a versatile rule such that $\mathsf{Max}(\mathsf{R}) \subseteq \{w_1, \ldots, w_n\}$;
- $\theta = \mathsf{mgu}(\langle w, w_1, \ldots, w_n\rangle, \langle v, u_1, \ldots, u_n\rangle)$, $u_1, \ldots, u_n \in \mathsf{St}(T) \setminus \mathcal{X}$, and $v \notin \mathcal{X}$;
- T' is a left-hand side of a deducibility constraint such that $T' \subsetneq T$.

Figure 2.3 - Simplification rules for good inference systems

- *π is reduced to a leaf or $\mathtt{last}(\pi)$ is an instance of a decomposition rule and we have that $\mathtt{step}(\pi) \subseteq \mathsf{St}(T)$;*
- *$\mathtt{last}(\pi)$ is an instance of a versatile rule and $\mathtt{step}(\pi') \subseteq \mathsf{St}(T)$ for any strict subproof π' of π.*

This is proved by observing that any proof in which a maximal conclusion is also a maximal premise of the next rule can be simplified, according to the definition of good inference systems.

2.2.2 The finite case

In this section, we consider good inference systems that only contain a finite number of inference rules. We show that we can solve deducibility constraints in such a way that *we do not miss any solution* (as in [CLCZ10]). The basic idea of the simplification rules described in Figure 2.3 is very straightforward, and that is what makes it appealing: we simply guess the last step of the proof, performing a backwards proof search together with narrowing the variables of the constraint. Assume that $\mathsf{R} = \dfrac{u_1 \ \ldots \ u_n}{u}$ is guessed as the last rule in the proof of $T\sigma \vdash v\sigma$, we simply perform:

$$\mathcal{C} \wedge T \overset{?}{\vdash} v \rightsquigarrow \mathcal{C}\theta \wedge T\theta \overset{?}{\vdash} u_1\theta \wedge \ldots \wedge T\theta \overset{?}{\vdash} u_n\theta$$

where $\theta = \mathsf{mgu}(u, v)$. This hardly terminates, even for very simple proof systems and ground goals. First, we do not aim at explicitly enumerating all possible solutions, but only compute *solved forms*, that are a convenient representation of all these solutions. Second, to avoid

non-terminating behaviours, we use locality (Lemma 2.1). We control the application of such rules, roughly requesting that maximal premises are subterms of T. Actually, we lifted this result to the non-ground case.

The purpose of each simplification rule is quite simple. Only the rule $\mathsf{R}_{\mathsf{Decompo}}$ deserves some explanation. We guessed here a versatile or a decomposition rule. The premises $w_1, \ldots, w_n$ will be those whose instances correspond to a term in $\mathsf{St}(T)\sigma$. We can guess the corresponding terms in $\mathsf{St}(T)$, namely $u_1, \ldots, u_n$. The other premises (that are then subterms in the substitution part) are constrained to be proved with strictly less hypotheses. We will show that this can always be assumed, hence that we get completeness.

The simplification rules transform a constraint system into a constraint system. Moreover, we have that:

Proposition 2.1 *[18] Let $\mathcal{C}$ be an unsolved constraint system.*

- *(Soundness) If $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ for some constraint system $\mathcal{C}'$ and some substitution σ, and if θ is a solution of $\mathcal{C}'$, then $\sigma\theta$ is a solution of $\mathcal{C}$.*
- *(Completeness) If θ is a solution of $\mathcal{C}$, then there exist a solved constraint system $\mathcal{C}'$ and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\mathcal{C} \rightsquigarrow_\sigma^* \mathcal{C}'$ and θ' is a solution of $\mathcal{C}'$.*

In order to get termination, we add some control on the simplification rules. Altogether, this gives us a procedure to derive a complete and effective representation of all solutions of a constraint system. This works for any finite good inference system that satisfies the additional syntactic conditions mentioned on page 34.

2.2.3 The infinite case: blind signatures

Consider a toy protocol in which A generates a nonce n, sends it to B, then B sends back this nonce signed with its private key k and finally A checks that the message he receives is $\mathsf{sign}(n, k)$. The possible traces obtained in any successful session of this protocol, between the two parties a, b are sequences of four messages:

$$n,\ x,\ \mathsf{sign}(x, k),\ \mathsf{sign}(n, k)$$

where n is the message sent by a, x is the message received by b, $\mathsf{sign}(x, k)$ is the message sent by b and $\mathsf{sign}(n, k)$ is the message received by a.

In an honest run, $x = n$. There are however other possible bindings of x, all yielding a valid trace. Actually, the only constraints that x must satisfy are:

$$\{n\} \overset{?}{\vdash} x \quad \text{and} \quad \{n,\ \mathsf{sign}(x, k)\} \overset{?}{\vdash} \mathsf{sign}(n, k)$$

Intuitively, the attacker should be able to construct x and to construct back $\mathsf{sign}(n, k)$ from n and $\mathsf{sign}(x, k)$. The set of such possible messages x includes n, but also $\mathsf{blind}(n, n)$ for instance, since the attacker can unblind $\mathsf{sign}(\mathsf{blind}(n, n), k)$ and get $\mathsf{sign}(n, k)$. Actually, the set of possible messages x satisfying the above constraints is $\mathcal{B}d(\{n\}, n)$ where $\mathcal{B}d(T, u)$ denotes the least set S of terms that contains u and such that $\mathsf{blind}(s, v) \in S$ when $s \in S$ and $T \vdash v$. In other words, we have that

$$\mathcal{B}d(T, u) = \{u\} \cup \{\mathsf{b}_k(u, v_1, \ldots, v_k) \mid k \in \mathbb{N} \text{ and } T \vdash v_i \text{ for each } 1 \leq i \leq k\}.$$

For each $t \in \mathcal{B}d(\{n\}, n)$, the attacker can compute $\mathsf{sign}(n, k)$ using one instance of the rule scheme (UB) and these are the only ways to satisfy the last constraint. We wish to use a single constraint solving step for all these possible final inference rules, hence we introduce an appropriate abstraction, enriching the syntax of constraint systems with *membership constraints*.

Extended constraint systems. We now consider two kinds of elementary constraints:

- a *deducibility constraint* is a constraint of the form $T \stackrel{?}{\vdash} u$, whereas
- a *membership constraint* is a constraint of the form $v \stackrel{?}{\in} \mathcal{B}d(T, u)$.

In both cases, T is a finite set of terms and u, v are terms. The set of terms T is called the *associated set of terms* of the elementary constraint. The notion of constraint system has to be extended accordingly. *Pure constraint systems* are constraint systems that only contain deducibility constraints (as defined in Definition 2.1). A solution is a substitution σ such that:

- $T\sigma \vdash u\sigma$ for each constraint $T \stackrel{?}{\vdash} u$ that occurs in $\mathcal{C}$; and
- $u\sigma \in \mathcal{B}d(T\sigma, v\sigma)$ for each constraint $u \stackrel{?}{\in} \mathcal{B}d(T, v)$ that occurs in $\mathcal{C}$.

Our aim is to design a set of simplification rules that rewrite any pure constraint system into a finite set of solved forms, which are more convenient representation of the same set of solutions. We have to extend our notion of solved form: now a deducibility constraint $T_i \stackrel{?}{\vdash} x_i$ can be associated with a membership constraint of the form $x_i \stackrel{?}{\in} \mathcal{B}d(T_i, v_i)$. Unfortunately, solved forms do not necessarily have a solution. Actually, our simplification rules will satisfy an additional invariant (called *well-formedness*) ensuring the existence of solutions.

Our simplification rules. The simplification rules are displayed in Figure 2.4. There is one simplification rule for each deduction rule. Note that the simplification rule $\mathsf{R}_{\mathsf{bdsgn}}$ is the simplification rule associated to the rule scheme (UB). This is where the membership constraints are introduced. The three last simplification rules allow us to cope with the membership constraints in order to reach a solved form. Moreover, we have some additional rules (that are not described here) that consist of unfolding the left-hand side of a membership constraint until a variable is reached. These rules simply reflect the semantics of $u \stackrel{?}{\in} \mathcal{B}d(T, v)$. We have shown soundness and completeness of our set of simplification rules.

Our decision procedure. Let $\mathcal{C}_0$ be a pure constraint system.

1. Guess a set of equalities E between subterms of $\mathcal{C}_0$. Solve E and let $\theta_E = \mathsf{mgu}(E)$ (if there is no solution, then return $\bot$).

2. Apply non-deterministically the simplification rules (Figure 2.4) on each $\mathcal{C}\theta_E$ until either a solved form is reached or a loop is detected (*i.e.* $\mathcal{C} \rightsquigarrow^n \mathcal{C}$ with $n \geq 1$), in which case we return $\bot$.

$$\mathsf{R}_{\mathsf{ax}}: \quad T \stackrel{?}{\vdash} u \rightsquigarrow \top \qquad \text{if } u \in T \setminus \mathcal{X}$$

$$\mathsf{R}_{\mathsf{triv}}: \quad T \stackrel{?}{\vdash} x \wedge T' \stackrel{?}{\vdash} x \rightsquigarrow T \stackrel{?}{\vdash} x \qquad \text{if } T \subseteq T'$$

$$\mathsf{R}_{\mathsf{f}}: \quad T \stackrel{?}{\vdash} \mathsf{f}(t_1, \ldots, t_n) \rightsquigarrow T \stackrel{?}{\vdash} t_1, \ldots, T \stackrel{?}{\vdash} t_n \qquad \mathsf{f} \in \{\mathsf{sign}, \mathsf{blind}\}$$

$$\mathsf{R}_{\mathsf{bd}}: \quad T \stackrel{?}{\vdash} v \rightsquigarrow T \stackrel{?}{\vdash} \mathsf{blind}(v, u) \wedge T \stackrel{?}{\vdash} u \qquad \text{if } \mathsf{blind}(v, u) \in \mathsf{St}(T)$$

$$\mathsf{R}_{\mathsf{get}}: \quad T \stackrel{?}{\vdash} v \rightsquigarrow T \stackrel{?}{\vdash} \mathsf{sign}(v, u) \qquad \text{if } \mathsf{sign}(v, u) \in \mathsf{St}(T)$$

$$\mathsf{R}_{\mathsf{bdsgn}}: \quad T \stackrel{?}{\vdash} \mathsf{sign}(v, u) \rightsquigarrow T \stackrel{?}{\vdash} \mathsf{sign}(w, u) \wedge w \stackrel{?}{\in} \mathcal{B}d(T, v) \qquad \text{if } \mathsf{sign}(w, u) \in \mathsf{St}(T)$$

$$\mathsf{R}_{\mathsf{A}}: \quad T \stackrel{?}{\vdash} x \wedge x \stackrel{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow T \stackrel{?}{\vdash} x \wedge T \stackrel{?}{\vdash} v \wedge x \stackrel{?}{\in} \mathcal{B}d(T, v) \qquad \text{if } T \subsetneq T'$$

$$\mathsf{R}_{\mathsf{B}}: \quad T \stackrel{?}{\vdash} x \wedge x \stackrel{?}{\in} \mathcal{B}d(T', v) \rightsquigarrow T \stackrel{?}{\vdash} x \wedge T \stackrel{?}{\vdash} w \wedge x \stackrel{?}{\in} \mathcal{B}d(T, w) \wedge w \stackrel{?}{\in} \mathcal{B}d(T', v)$$
$$\text{if } T \subsetneq T' \text{ and } w \in \mathsf{St}(T)$$

$$\mathsf{R}_{\mathsf{C}}: \quad T \stackrel{?}{\vdash} x \wedge x \stackrel{?}{\in} \mathcal{B}d(T, v) \wedge x \stackrel{?}{\in} \mathcal{B}d(T, v') \rightsquigarrow T \stackrel{?}{\vdash} x \wedge x \stackrel{?}{\in} \mathcal{B}d(T, v) \wedge v \stackrel{?}{\in} \mathcal{B}d(T, v')$$
$$\text{if } x \text{ does not occur in any constraint having as associated set } T' \subsetneq T.$$

Figure 2.4 - Simplification rules for blind signatures

Considering all possible non-deterministic choices that do not yield $\bot$, the procedure computes a finite set of pairs $(E_i, \mathcal{C}_i)$ such that every $\mathcal{C}_i$ is in solved form.

Theorem 2.2 *[53] The procedure described above is:*

- *sound,* i.e. *for any σ_i solution of an output pair $(E_i, \mathcal{C}_i)$, we have that $\theta_{E_i}\sigma_i$ is a solution of $\mathcal{C}_0$;*
- *complete,* i.e. *for any solution σ of $\mathcal{C}_0$, there exists an output pair $(E_i, \mathcal{C}_i)$, and a solution σ_i of $\mathcal{C}_i$ such that $\sigma = \theta_{E_i}\sigma_i$;*
- *terminating.*

We have applied the same technique to derive a decision procedure for an inference system modelling symmetric encryption using an ECB encryption mode (see [53] for more details).

2.3 Ad hoc routing protocols

Mobile ad hoc networks consist of mobile wireless devices which autonomously organise their communication infrastructure: each node provides the function of a router and relays packets on paths to other nodes. Finding these paths is a crucial functionality of any ad hoc network. Specific protocols, called *routing protocols*, are designed to ensure this functionality known as *route discovery*.

Initial work on routing in ad hoc networks has considered only the problem of providing efficient mechanisms for finding paths, without considering security issues. Recent research

has recognized that this assumption is unrealistic and that attacks can be mounted [HPJ05, NH06, BV04]. Since an adversary can easily paralyse the operation of a whole network by attacking the routing protocol, it is crucial to prevent malicious nodes from compromising the discovered routes. Since then, secure versions of routing protocols have been developed to ensure that mobile ad hoc networks can work even in an adversarial setting [HPJ05, PH02]. Those routing protocols use cryptographic mechanisms such as encryption, signature, MAC, in order to prevent a malicious node to insert and delete nodes inside a path.

While key-exchange protocols are well-studied, there are very few attempts to develop formal techniques allowing an automated analysis of secure routing protocols. Those protocols indeed involve several subtleties that cannot be reflected in existing work. For example, the underlying network topology is crucial to define who can receive the messages sent by a node and the attacker is localised at some specific nodes (possibly several nodes). Moreover, the security properties include *e.g.* the validity of a route, which differs from the usual secrecy and authenticity properties. We show in this section how constraint solving techniques can be adapted to analyse secure routing protocols (see [16] for more details).

Related work. Recently, some frameworks have been proposed to model wireless communication and/or routing protocols in a more accurate way. For example, S. Yang and J. Baras [YB03] provide a first symbolic model for routing protocols based on strand spaces, modelling the network topology but not the cryptographic primitives that can be used for securing communications. They also implement a semi-decision procedure to search for attacks. S. Nanz and C. Hankin [NH06] propose a process calculus to model the network topology and broadcast communications. They also propose a decision procedure but for an attacker that is already specified by the user. This allows to check security only against fixed, known in advance scenarios. The model proposed in this paper is inspired from their work, adding in particular a logic for specifying the tests performed at each step by the nodes on the current route and to specify the security properties. P. Schaller *et al* [SSBC09] propose a symbolic model that allows an accurate representation of the physical properties of the network, in particular the speed of the communication. This allows in particular to study distance bounding protocols. Several security proofs are provided for some fixed protocols, formalized in Isabelle/HOL.

2.3.1 Model for ad hoc routing protocols

In [16], we designed a calculus, inspired from CBS# [NH06], which allows mobile wireless networks and their security properties to be formally described and analysed. We propose in particular a logic to express the tests performed by the nodes at each step. It allows for example checking whether a route is "locally" valid, given the information known by the node.

Several calculi already exist to model security protocols (*e.g.* [AF01] for which a brief presentation is proposed in Chapter 4). However, for our purpose, a node, *i.e.* a process, has to perform some specific actions that can not be easily modelled in such calculi. For instance, a node stores some information, *e.g.* the content of its routing table. We also need to take into account the network topology and to model broadcast communication. Such features can not be easily modelled in these calculi. Moreover, sometimes nodes perform some sanity checks on the routes they receive, such as neighbourhood properties. We briefly present below the main features of our calculus together with its informal semantics.

Neighbourhood properties. Secured routing protocols typically perform some checks on the message they received before accepting it. Thus we will typically consider the logic $\mathcal{L}_{\mathsf{route}}$ defined by the grammar given in Figure 2.5.

$\Phi :=$	formula
$\mathsf{check}(a, b)$	neighbourhood of two nodes
$\mathsf{checkI}(c, l)$	local neighbourhood of a node in a list
$\mathsf{route}(l)$	validity of a route
$\mathsf{loop}(l)$	existence of a loop in a list
$\Phi_1 \wedge \Phi_2$	conjunction
$\Phi_1 \vee \Phi_2$	disjunction
$\neg\Phi$	negation

Figure 2.5 - The logic $\mathcal{L}_{\mathsf{route}}$

Given an undirected graph $G = (V, E)$ the formula $\mathsf{check}(a, b)$ expresses that $(a, b) \in E$, $\mathsf{checkI}(c, l)$ allows us to express that c appears exactly once in l and that the neighbours of c in l are neighbours of c in the graph G. The semantics of the other constructions is straightforward.

Processes. The intended behaviour of each node of the network can be modelled by a *process* defined by the grammar presented in Figure 2.6. Our calculus is parametrised by a set $\mathcal{L}$ of formulas.

$P, Q :=$	processes	
0		null process
$\mathtt{out}(u).P$		emission
$\mathtt{in}\ u[\Phi].P$		reception, $\Phi \in \mathcal{L}$
$\mathtt{store}(u).P$		storage
$\mathtt{read}\ u\ \mathtt{then}\ P\ \mathtt{else}\ Q$		reading
$\mathtt{if}\ \Phi\ \mathtt{then}\ P\ \mathtt{else}\ Q$		conditional, $\Phi \in \mathcal{L}$
$P \mid Q$		parallel composition
$!P$		replication
$\mathtt{new}\ m.P$		fresh name generation

Figure 2.6 - Processes

The process $\mathtt{out}(u).P$ emits u and then behaves like P. The process $\mathtt{in}\ u[\Phi].P$ expects a message m of the form u such that Φ is true and then behaves like $P\sigma$ where $\sigma = \mathsf{mgu}(m, u)$. If Φ is the true formula, we simply write $\mathtt{in}\ u.P$. The process $\mathtt{store}(u).P$ stores u in its storage list and then behaves like P. The process $\mathtt{read}\ u\ \mathtt{then}\ P\ \mathtt{else}\ Q$ looks for a message of the form u in its storage list and then, if such an element m is found, it behaves like $P\sigma$ where $\sigma = \mathsf{mgu}(m, u)$. If no element of the form u is found, then it behaves like Q. Sometimes, for the sake of clarity, we will omit the null process. We also omit the else part when $Q = 0$.

Network topology. Each process is located at a specified node of the network. Unlike classical Dolev-Yao model, the attacker does not control the entire network but can only interact with his neighbours. More specifically, we assume that the topology of the network is represented by giving an undirected graph $G = (V, E)$, where an edge in the graph models the fact that two nodes are neighbours. We also assume that we have a set of nodes $\mathcal{M}$ that are controlled by the attacker. These nodes are then called *malicious*. Our model is not restricted to a single malicious node. Our results allow us to consider the case of several compromised nodes that collaborate by sharing their knowledge. However, it is well-known that the presence of several colluding malicious nodes often yields straightforward attacks [HPJ06, LPM+05].

A *concrete configuration* of the network is a tuple $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ where:

- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$. The expression $\lfloor P \rfloor_n$ represents the process P located at node $n \in V$. We will write $\lfloor P \rfloor_n \cup \mathcal{P}$ instead of $\{\lfloor P \rfloor_n\} \cup \mathcal{P}$.
- $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in V$ and t a ground term. The expression $\lfloor t \rfloor_n$ represents the fact that the node n has stored the term t.
- $\mathcal{I}$ is a set of terms representing the messages seen by the attacker.

Each honest node broadcasts his messages to all his neighbours. To capture malicious behaviours, we allow the nodes controlled by the attacker to send messages only to some specific neighbour. The communication system is formally defined by rules that are parametrised by the underlying graph G and the set of malicious nodes $\mathcal{M}$. We denote by $\rightarrow_{G,\mathcal{M}}$ the relation between configurations induced by these rules. The relation $\rightarrow^*_{G,\mathcal{M}}$ is the reflexive and transitive closure of $\rightarrow_{G,\mathcal{M}}$. We may write $\rightarrow$, $\rightarrow_G$, $\rightarrow_{\mathcal{M}}$ instead of $\rightarrow_{G,\mathcal{M}}$ when the underlying network topology G or the underlying set $\mathcal{M}$ is clear from the context.

Note that in case we assume that there is a single malicious node and each honest node is connected to it, we retrieve the model where the attacker is assumed to control all the communications. As usual, an attack is defined as a reachability property.

Definition 2.4 *Let G be a graph and $\mathcal{M}$ be a set of nodes. There is an $\mathcal{M}$-*attack *on a configuration with a hole $(\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ for the network topology G and the formula Φ if there exist $n, \mathcal{P}', \mathcal{S}', \mathcal{I}'$ such that:*

$$(\mathcal{P}[\mathit{if}\ \Phi\ \mathit{then}\ \mathit{out}(\mathsf{error})]; \mathcal{S}; \mathcal{I}) \rightarrow^*_{G,\mathcal{M}} (\lfloor \mathit{out}(\mathsf{error}) \rfloor_n \cup \mathcal{P}', \mathcal{S}', \mathcal{I}')$$

where error *is a special symbol not occurring in the configuration $(\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$.*

2.3.2 Application: the SRP protocol

As an illustrative example, we consider the Secure Routing Protocol SRP introduced in [PH02], assuming that each node already knows his neighbours (running *e.g.* some neighbour discovery protocol). SRP is not a routing protocol by itself, it describes a generic way for securing source-routing protocols. In this section, we model its application to the DSR protocol [JMB01]. DSR is a protocol which is used when an agent S (the source) wants to communicate with another agent D (the destination), which is not his immediate neighbour. In an ad hoc network, messages can not be sent directly to the destination, but have to travel along a path of nodes.

Attacker capabilities. As explained in Section 2.1.1, the ability of the attacker is modelled by a deduction system. In order to model routing protocols, we consider a special sort loc for the nodes of the network. We assume that names and variables are given with sorts. We also assume an infinite subset $\mathcal{N}_{\mathsf{loc}}$ of names of sort loc. To model SRP, we will consider the specific signature $(\mathcal{S}_1, \mathcal{F}_1)$ defined by $\mathcal{S}_1 = \{\mathsf{loc}, \mathsf{lists}, \mathsf{terms}\}$ and $\mathcal{F}_1 = \{\mathsf{hmac}, \langle\ \rangle, ::, \bot, \mathsf{senc}\}$, with the following arities:

- $\mathsf{hmac}, \mathsf{senc}, \langle \_, \_ \rangle : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$,
- $:: : \mathsf{loc} \times \mathsf{lists} \to \mathsf{lists}$,
- $\bot :\to \mathsf{lists}$.

The sort lists represents lists of terms of sort loc. The symbol :: is the list constructor. $\bot$ is a constant representing an empty list. The term $\mathsf{hmac}(m, k)$ represents the keyed hash message authentication code computed over message m with key k. The other function symbols have been already defined. We write $\langle t_1, t_2, t_3 \rangle$ for the term $\langle t_1, \langle t_2, t_3 \rangle \rangle$, and $[t_1; t_2; t_3]$ for $t_1 :: (t_2 :: (t_3 :: \bot))$. We consider the inference system described in Figure 2.1 with some additional inference rules to deal with lists and hmacs.

$$\frac{x \qquad x_l}{x :: x_l} \qquad \frac{x :: x_l}{x} \qquad \frac{x :: x_l}{x_l} \qquad \frac{x_1 \qquad x_2}{\mathsf{hmac}(x_1, x_2)}$$

Processes. To discover a route to the destination, the source constructs a request packet and broadcasts it to its neighbours. The request packet contains its name S, the name of the destination D, an identifier of the request id, a list containing the beginning of a route to D, and a hmac computed over the content of the request with a key K_{SD} shared by S and D. It then waits for an answer containing a route to D with a hmac matching this route, and checks that it is a plausible route by checking that the route does not contain a loop and that his neighbour in the route is indeed a neighbour of S in the network.

Let $S, D, \mathsf{req}, \mathsf{rep}, id, K_{SD}$ be names ($S, D \in \mathcal{N}_{\mathsf{loc}}$) and x_L be a variable of sort lists. The process executed by a node S initiating the search of a route towards a node D is:

$$P_{\mathsf{init}}(S, D) = \mathtt{new}\ id.\mathtt{out}(u_1).\mathtt{in}\ u_2[\Phi_S].0$$

where:

$$\begin{cases} u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, \mathsf{hmac}(\langle \mathsf{req}, S, D, id \rangle, K_{SD}) \rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_L, \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, x_L \rangle, K_{SD}) \rangle \\ \Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg \mathsf{loop}(x_L). \end{cases}$$

The names of the intermediate nodes are accumulated in the route request packet. Intermediate nodes relay the request over the network, except if they have already seen it. An intermediate node also checks that the received request is locally correct by verifying whether the head of the list in the request is one of its neighbours. Below, $V \in \mathcal{N}_{\mathsf{loc}}$, x_S, x_D and x_a are variables of sort loc whereas x_r is a variable of sort lists and x_{id}, x_m are variables of sort terms. The process executed by an intermediary node V when forwarding a request is as follows:

$$P_{\mathsf{req}}(V) = \mathtt{in}\ w_1[\Phi_V].\mathtt{read}\ t\ \mathtt{then}\ 0\ \mathtt{else}\ (\mathtt{store}(t).\mathtt{out}(w_2))$$

where:

$$\begin{cases} w_1 = \langle \mathsf{req}, x_S, x_D, x_{id}, x_a :: x_r, x_m \rangle \\ \Phi_V = \mathsf{check}(V, x_a) \\ t = \langle x_S, x_D, x_{id} \rangle \\ w_2 = \langle \mathsf{req}, x_S, x_D, x_{id}, V :: (x_a :: x_r), x_m \rangle \end{cases}$$

When the request reaches the destination D, it checks that the request has a correct hmac and that the first node in the route is one of his neighbours. Then, the destination D constructs a route reply, in particular it computes a new hmac over the route accumulated in the request packet with K_{SD}, and sends the answer back over the network.

The process executed by the destination node D is the following:

$$P_{\mathsf{dest}}(D, S) = \mathtt{in}\ v_1[\Phi_D].\mathtt{out}(v_2).0$$

where:

$$\begin{array}{l} v_1 = \langle \mathsf{req}, S, D, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{req}, S, D, x_{id} \rangle, K_{SD}) \rangle \\ \Phi_D = \mathsf{check}(D, x_a) \\ v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, \mathsf{hmac}(\langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l \rangle, K_{SD}) \rangle \end{array}$$

Then, the reply travels along the route back to S. The intermediate nodes check that the route in the reply packet is locally correct (that is that they appear once in the list and that the nodes before and after them are their neighbours) before forwarding it. The process executed by an intermediate node V when forwarding a reply is the following:

$$P_{\mathsf{rep}}(V) = \mathtt{in}\ w'[\Phi'_V].\mathtt{out}(w')$$

where:

$$\begin{cases} w' = \langle \mathsf{rep}, x_D, x_S, x_{id}, x_r, x_m \rangle \\ \Phi'_V = \mathsf{checkI}(V, x_r) \end{cases}$$

Security property. For the SRP protocol, the property we want to check is that the list of nodes obtained by the source through the protocol represents a path in the graph. We can easily encode this property by replacing the null process in $P_{\mathsf{init}}(S, D)$ by a hole, and checking whether the formula $\neg\mathsf{route}(x_L)$ holds. Let $P'_{\mathsf{init}}(S, D)$ be the resulting process.

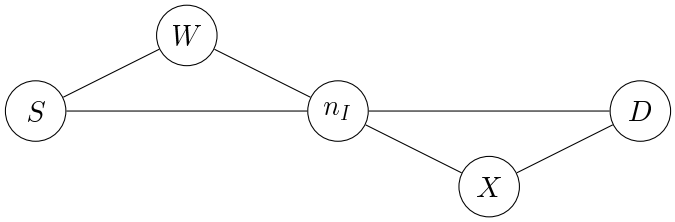
$$P'_{\mathsf{init}}(S, D) = \mathtt{new}\ id.\mathtt{out}(u_1).\mathtt{in}\ u_2[\Phi_S].P$$

where $P = \mathsf{if}\ \neg\mathsf{route}(x_L)\ \mathsf{then}\ \mathtt{out}(\mathsf{error})$.

Initial configuration. A typical initial configuration for the SRP protocol is

$$K_0 = (\lfloor P'_{\mathsf{init}}(S, D) \rfloor_S \mid \lfloor P_{\mathsf{dest}}(D, S) \rfloor_D; \emptyset; \mathcal{I}_0)$$

where both the source node S and the destination node D wish to communicate. We assume that each node has an empty storage list and that the initial knowledge of the attacker is given by a set of terms $\mathcal{I}_0$. A possible network configuration is modelled by the graph G_0 below. We assume that there is a single malicious node, *i.e.* $\mathcal{M}_0 = \{n_I\}$. The nodes W and X are two extra (honest) nodes. We do not assume that the intermediate nodes W and X execute the routing protocol. Actually, this is not needed to show that the protocol is flawed, and we want to keep this example as simple as possible.



Description of an attack. We recover the attack mentioned in [BV04] with the topology G_0, and from the initial configuration K_0. The attack scenario is the following. The source S sends a route request towards D. The request reaches the node n_I. Thus, the attacker receives the following message:

$$\langle \mathsf{req}, S, D, id, S :: \bot, \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle$$

The attacker then broadcasts the following message in the name of X:

$$\langle \mathsf{req}, S, D, id, [X; W; S], \mathsf{hmac}(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle$$

Since D is a neighbor of n_I, it will hear the transmission. In addition, since the list of nodes $[X; W; S]$ ends with X, which is also a neighbor of D, the destination D will process the request and will send the following route reply back to S:

$$\langle \mathsf{rep}, D, S, id, [X; W; S], \mathsf{hmac}(\langle \mathsf{rep}, D, S, id, [X; W; S]\rangle, K_{SD})\rangle$$

This reply will reach S through the malicious node n_I. More precisely, the attacker will send the reply to S in the name of W. Since W is a neighbor of S, the source will accept this reply which contains a false route.

In our framework, this attack can be described as follows:

$$\begin{array}{l} K_0 \to^* (\lfloor \mathtt{in}\ u_2[\Phi_S].P\rfloor_S \cup \lfloor \mathtt{out}(m').0\rfloor_D; \emptyset; \mathcal{I}) \\ \quad \to (\lfloor \mathtt{in}\ u_2[\Phi_S].P\rfloor_S \cup \lfloor 0\rfloor_D; \emptyset; \mathcal{I}') \\ \quad \to (\lfloor \mathsf{if} \neg \mathsf{route}([X; W; S])\ \mathsf{then}\ \mathtt{out}(\mathsf{error})\rfloor_S; \emptyset; \mathcal{I}') \\ \quad \to (\lfloor \mathtt{out}(\mathsf{error}).0\rfloor_S; \emptyset; \mathcal{I}') \end{array}$$

where $\begin{cases} m' = \langle \mathsf{rep}, D, S, id, [X; W; S], \mathsf{hmac}(\langle D, S, id, [X; W; S]\rangle, K_{SD})\rangle \\ \mathcal{I} = \mathcal{I}_0 \cup \{u_1\}, \text{and } \mathcal{I}' = \mathcal{I}_0 \cup \{u_1\} \cup \{m'\}. \end{cases}$

2.3.3 Decidability and complexity results

We assume the fixed signature $(\mathcal{S}_1, \mathcal{F}_1)$ (defined in Section 2.3.2) for list, concatenation, hmac and encryption together with its associated inference system. Simple properties like secrecy are undecidable when considering an unbounded number of role executions, even for classical protocols [DLMS99]. Since our class of processes encompasses classical protocols, the existence of an attack is also undecidable. In what follows, we thus consider a finite number of sessions, that is processes without replication. We say that a process is *finite* if it does not contain the replication operator. A concrete configuration $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ is said *initial* if K is ground, $\mathcal{P}$ is finite, $\mathcal{S}$ is a finite set of terms and $\mathcal{I} = \mathcal{N}_{|\mathsf{oc}} \cup \mathcal{I}'$ where $\mathcal{I}'$ a finite set of terms (the attacker is given all the node names in addition to its usual initial knowledge).

We show that accessibility properties are decidable for finite processes of our process algebra, which models secure routing protocols, for a bounded number of sessions. We actually provide two decision procedures, according to whether the network is *a priori* given or not. In case the network topology is not fixed in advance, our procedure allows us to automatically discover whether there exists a (worst-case) topology that would yield an attack.

Theorem 2.3 *[16] Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with an hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists a graph G such that there is an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Theorem 2.4 *[16] Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with an hole, G be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.*

Note that Theorem 2.3 does not imply Theorem 2.4 and reciprocally. Theorems 2.3 and 2.4 ensure in particular that we can decide whether a routing protocol like SRP can guarantee that any route accepted by the source is indeed a route (a path) in the network (which can be fixed by the user or discovered by the procedure).

In this setting, we have seen that groups of executions can be represented using constraint systems. However, we have to enrich constraint systems in order to cope with the formulas that are checked upon the reception of a message and also in order to cope with generalised disequality tests for reflecting cases where agents reject messages of the wrong form.

Extended constraint systems. We now have to consider three kinds of elementary constraints:

- a *deducibility constraint* is a constraint of the form $T \stackrel{?}{\vdash} u$;
- a *unification constraint* is a constraint of the form $v = u$;
- a *disequality constraint* is a constraint of the form $\forall X.\, v \neq u$;

where v, u are terms, T is a non empty set of terms, and X is a set of variables. The notion of constraint system has to be extended accordingly. A *constraint system* $\mathcal{C}$ is a finite conjunction of *constraints* of the form described above together with a formula Φ of $\mathcal{L}_{\mathsf{route}}$. Moreover, we assume that the elementary constraints in $\mathcal{C}$ can be ordered in such a way that monotony and origination hold.

A *solution* to a constraint system $\mathcal{C}$ for a graph G is a ground substitution θ such that $t\theta = u\theta$ for all $t = u \in \mathcal{C}$, $T\theta \vdash u\theta$ for all $T \stackrel{?}{\vdash} u \in \mathcal{C}$, and $\Phi\theta$ is evaluated to true. Lastly, for all $(\forall X.\, t \neq u) \in \mathcal{C}$, we have that $t\theta$ and $u\theta$ are not unifiable. Note that variables in X are bound and thus renamed before the application of the substitution θ.

Decision procedures. The (NP) decision procedures proposed for proving Theorems 2.3 and 2.4 involve several steps, with many common ingredients.

Step 1. First, we show that it is sufficient to decide whether there exists a sequence of symbolic transitions (and a graph G if G is not fixed) with a resulting symbolic configuration $(\lfloor \mathtt{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$ such that $\mathcal{C} \wedge u = \mathsf{error}$ admits a solution for the graph G. Since

processes contain no replication and involve communication between a finite number of nodes, it is possible to guess the sequence of symbolic transitions yielding an attack (by guessing also the edges between the nodes that are either in $\mathcal{M}$ or involved in a communication step) and the resulting configuration remains of size polynomially bounded by the size of the initial configuration. Moreover, any left-hand side of a deduction constraint in $\mathcal{C}$ is of the form $T \cup \mathcal{N}_{\mathsf{loc}}$ where T is a finite set of terms. It then remains to decide the existence of a solution for our class of constraint systems.

Step 2. It has been shown in [CZ06] and reviewed in Section 2.1.3 that the existence of a solution of a constraint system (with only deduction constraints) can be reduced to the existence of a solution of a *solved* constraint system, where right-hand sides of the constraints are variables only. We have extended this result to our extended notion of constraint systems, *i.e.* with disequality tests and formula of $\mathcal{L}_{\mathsf{route}}$.

Step 3. We then show how to decide the existence of a solution for a constraint system, where each deduction constraint is solved, that is of the form $T \stackrel{?}{\vdash} x$. This is not straightforward like in [CZ06] since we are left with (non solved) disequality constraints and formulas. The key step consists in showing that we can bound (polynomially) the size of the lists in a minimal attack.

We would like to emphasise that our model is not strictly dedicated to routing protocols but can be used to model many other classes of protocols. In particular, by considering a special network topology where the attacker is at the centre of the network, we retrieve the classical modelling where the attacker controls all the communications. We thus can model as usual all the key exchange and authenticity protocols presented *e.g.* in the Clark & Jacob library [CJ97]. Since we also provide each node with a memory, our model can also capture protocols where a state global to all sessions is assumed for each agent. It is typically the case of protocols where an agent should check that a key has not been already accepted in a previous session, in order to protect the protocol against replay attacks.

2.4 Conclusion and perspectives

In this chapter, we have demonstrated that constraint solving is a powerful technique to analyse security protocols. Moreover, this technique is quite flexible and can be extended in several ways. We will see in Chapter 6 that constraint solving is also suitable to deal with equivalence-based properties such as privacy-type security properties.

We claim that the key property of the inference system, which allows us to solve the deducibility constraints, is locality. Given an inference system, the general procedure then consists in completing the inference rules into a local inference system. When such a system is infinite, we need additional abstractions and simplification rules. We have shown in Section 2.2.3 that this is possible, in the case study of blind signatures. We demonstrated that the method is general enough, by giving another example of application, an inference system modelling symmetric encryption using an ECB encryption mode. It remains to provide with a general way of abstracting some classes of infinite inference systems that would be amenable to deducibility constraint solving. There are a few specialized theorem provers that are relying on constraint solving techniques [Cor06, Mil09, BMV05, Tur06]. But the recent

theoretical advances in this area may, in the next few years, yield new tools based on these techniques.

To our knowledge, the result presented in Section 2.3 is the first decidability and complexity result for routing protocols, for arbitrary attackers and network topologies. Moreover, since we reuse existing techniques on solving constraint systems, our decision procedure seems amenable to implementation, re-using existing tools. Recently, an analysis using the AVISPA tool (a tool dedicated to the formal analysis of classical cryptographic protocols) has been conducted on two routing protocols, namely ARAN and endairA [BMV10], yielding encouraging results.

We have seen that routing protocols have several specificities (*e.g.* neighbourhood properties, underlying topology, ...). Sometimes, secured versions of routing protocols [BV04, HPJ05, FGML09] also require the nodes (typically the node originating the request) to check the validity of the route they receive. This is usually performed by checking that each node has properly signed (or MACed) some part of the route, the whole incoming message forming a chain where each component is a contribution from a node in the path. Other examples of protocols performing recursive operations are certification paths for public keys (*e.g.* X.509 certification paths [HFP98]) or right delegation in distributed systems [Aur99]. For verifying security of protocols with recursive tests (for a bounded number of sessions), it seems possible to reuse the setting of constraint systems and add membership constraints to test that a part of an incoming message belongs to a recursive language. Of course, we have then to revisit the simplification rules to get a decision procedure.

Ad hoc networking is a new area in wireless communications that is attracting the attention of many researchers for its potential to provide ubiquitous connectivity without the assistance of any fixed infrastructure. An important aspect that is not taken into account in most of the analysis is mobility. It is crucial to take this feature into account especially in some applications such as vehicle-to-vehicle communications where routing protocols have to work in presence of fast moving nodes.

Chapter 3

Verification of security APIs

Contents

In contrast to the case of a bounded number of sessions in which decidability results for verifying cryptographic protocols could be obtained (see Chapter 2), the case of an unbounded number of sessions is well-known to be undecidable [DLM04]. Despite this negative result, formal methods have proved their usefulness in the rigorous analysis of security protocols even in presence of an unbounded number of sessions. In this chapter, we will concentrate on security APIs for tamper resistant devices. A security API can be seen as a collection of protocols. Therefore, methods developed for security-protocol analysis can be used for analysing such a device. For instance, the CL-Atse tool [Tur06] has been used in the analysis of the key management subset of the IBM 4758 API [CKS07]. We also propose an analysis of the TPM API in Section 3.2 with the ProVerif tool [Bla01]. However, it is well-admitted that security APIs have some specificities that prevent us sometimes to use tools designed for classical cryptographic protocols. After recalling some of these specificities (see Section 3.1),

we propose a formal model that is suitable to represent security APIs and that allows us to analyse PKCS#11. Lastly, we will propose an extension of the Dolev-Yao intruder model to capture a new kind of attack, called *key conjuring* (see Section 3.4).

3.1 Background

In this section, we first explain what a security API is. Then, we highlight the specificities of the security APIs that explain why in general a security protocol analyser produces poor results on this kind of applications.

3.1.1 Security APIs

The purpose of a security application programming interface (API) is to allow untrusted code to access sensitive resources in a secure way. They typically arise in systems where certain critical fragments of a program are executed on some tamper resistant device (TRD), such as a USB security key or a hardware security module (HSM). Hardware security modules, for example, are widely used in systems such as electronic payment and automated teller machine (ATM) networks. They typically consist of a cryptoprocessor and a small amount of memory inside a tamper proof enclosure. They are designed so that should an intruder open the casing or insert probes to try to read the memory, it will auto-erase in a matter of nanoseconds. Security APIs for tamper resistant hardware devices typically manage keys by keeping a secret master key inside the device. This key is then used to encrypt all the working keys used for operational functions, so that they can be securely stored outside the device. These keys can then only be used by sending them back into the HSM, together with some other data, under the HSM's application programming interface.

The API will typically consist of a set of functions designed to facilitate the secure generation, storage, use and destruction of cryptographic keys. One can think of the API as defining a number of two-party protocols, each describing an exchange between the tamper resistant device and the host machine. However there are some important differences between security APIs and more classical cryptographic protocols.

3.1.2 Some specificities

A security API can be seen as a collection of protocols which may be called in any order. However, some specificities, listed below, prevent us to use existing verification tools that have been designed to analyse more standard cryptographic protocols, *e.g.* key establishment protocols, authentication protocols, ...

Security policy. Security APIs are designed to ensure a policy, *i.e.* no matter what commands are received from the untrusted code, certain security properties will continue to hold, *e.g.* the secrecy of a sensitive cryptographic keys. In general, the security policy is not clearly specified (*e.g.* TPM [Tru07]) or is well-known to be not satisfied by the standard. In this last case, it is up to the developer to restrict the functionalities provided by the standard to obtain a secure device (*e.g.* PKCS #11 [RSA04]). In any case, this leaves the developer of an application in a confusing position. Indeed, he has to ensure some properties for his application without knowing precisely what is the security policy guaranteed by the APIs.

Key table. The aim of a tamper resistant device such as an HSM or a security USB key is to store cryptographic keys and other sensitive data in its shielded memory. Then, objects such as keys, are referenced in the APIs via *handles*, which can be thought of as pointers to the objects. In general, the value of the handle does not reveal any information about the actual value of the object. This key table is an important aspect that we have to take into account to analyse security APIs.

Non-monotonic global state. One of the difficulties in reasoning about security APIs is the need for *non-monotonic global states*. If the tamper resistant device is in a certain state s, and then a command is successfully executed, then typically the TRD ends up in a state $s' \neq s$. Commands that require it to be in the previous state s will no longer work. Although it is in general possible to encode the state changes in existing verification tools, the tools perform poorly on the resulting specification. For instance, in the ProVerif tool, private channels could be used to represent the state changes. However, the abstraction of private channels that ProVerif makes prevents it from being able to verify the correctness of the resulting specification. Security API models typically contain non-monotonic global state, which must be modelled accurately to get reasonable precision (*i.e.* not too many false attacks).

Intruder model. The intruder model most commonly used for analysing security protocols in symbolic approaches is the classical Dolev-Yao attacker (see Chapter 2). To identify how the attacker should be modelled to analyse security APIs, we have to understand what attacks concern HSM manufacturers (and their customers) in practice. Actually, some attacks are not captured by the standard Dolev-Yao model. For instance, one technique used by attackers attempting to breach security is to try calling APIs functions with random values in the place of encrypted keys, to see if they are allowed to pass, or whether the device signals an error. This process is known as *key conjuring* [Bon01]. Learning the encrypted value of a key might not seem useful, but several attacks have been presented that leverage this trick in order to compromise the security of an API [Bon01, Clu03a, CB03]. Hence, the analysis of security APIs involves adapting Dolev-Yao style protocol analysis techniques [DY81] to capture feasible brute-force attacks such as the one described above.

3.2 A formal analysis of authentication in the TPM

In this section, we show through a real case study how a security protocol analyser can be used to analyse a security API (more details about this work can be found in [14]). Because of the specificities mentioned in Section 3.1.2, some encodings are needed to model for instance the memory of the HSM. Nevertheless, in the case of the TPM applications, this allows one to obtain some interesting results. An important aspect of our work was to be able to find some suitable encodings and also to identify the generic security properties that are supposed to be achieved by the TPM (independently of the underlying application).

Several papers have appeared describing systems that leverage the TPM to create secure applications, but most of these assume that the TPM API correctly functions and provides the high-level security properties required [DFGK09, GSS+07]. Lower level analyses of the TPM API are more rare. Coker *et al.* discuss such work, but the details of the model remain classified [CGL+10]. Gürgens *et al.* [GRS+07] describe an analysis of the TPM API using

finite state automata. They formalise security as secrecy of certain terms in the model, giving examples of concrete scenarios where these secrets must be protected. They show how an attacker can in some circumstances illegitimately obtain a certificate for a TPM key of his choice. Their attack is not specific to the scenario they studied. Actually, this attack comes from the fact that the generic injective agreement property that we propose does not hold on the TPM CertifyKey command (see Figure 3.1).

Other attacks on the TPM found without the aid of formal methods include offline dictionary attacks on the passwords or authdata used to secure access to keys [CR08a], and attacks exploiting the fact that the same authdata can be shared between users [CR09]. Both of these could be detected in our formal model by small adjustments.

3.2.1 A brief 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. TPM keys are arranged in a tree structure, with the *storage root key* at its root. To each TPM key is associated a 160-bit string called *authdata*, which is analogous to a password that authorises the use of the key. A user can use a key loaded in the TPM through the interface provided by the device. 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*. This mechanism provides a way to exchange nonces that are then used to ensure freshness of the messages exchanged in a command.

To use a TPM key, it must first be created (via the command CreateWrapKey) and then loaded (via the command LoadKey2). Actually, LoadKey2 takes as argument a wrap created by the command CreateWrapKey, and returns a handle, that is, a pointer to the key stored in the TPM memory. Commands that use the loaded key refer to it by this handle. Since LoadKey2 involves a decryption by the parent key, it requires the parent key to be loaded and it requires an authorisation HMAC based on the parent key authdata. Once loaded, a key can be used, for example to encrypt or decrypt data, or to sign data. As an illustrative example, we describe the command CertifyKey in more detail.

Command CertifyKey. This command requires two key handles as arguments, representing the certifying key and the key to be certified, and two corresponding authorisation HMACs. It returns the certificate. Assume that the key pair $(sk_1, \mathsf{pk}(sk_1))$ (resp. $(sk_2, \mathsf{pk}(sk_2))$) is already loaded inside the TPM with $auth_1$ (resp. $auth_2$) as an associated authdata. We also assume that the access to these keys is possible through the handles handle_1 and handle_2. Lastly, we assume that two authorisation sessions are already established with the nonces ne_1 and ne_2 respectively. The CertifyKey command can be informally described as follows:

$$\begin{array}{rcl} n, no_1, no_2, & & \mathtt{new}\ Ne_1, \mathtt{new}\ Ne_2 \cdot \\ \mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, ne_1, no_1 \rangle) & & Ne_1, Ne_2, \mathrm{certif} \\ \mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, ne_2, no_2 \rangle) & \rightarrow & \mathsf{hmac}(auth_1, \langle \mathsf{cfk}, n, Ne_1, no_1, \mathrm{certif} \rangle) \\ \mathsf{handle}_1, \mathsf{handle}_2 & & \mathsf{hmac}(auth_2, \langle \mathsf{cfk}, n, Ne_2, no_2, \mathrm{certif} \rangle) \end{array}$$

where $\mathrm{certif} = \mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$.

The 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 (needed for the authorisation session

mechanism) that he gives to the TPM together with the two authorisation HMACs. The TPM returns the certificate certif. Two authentication HMACs are also built by the TPM to accompany the answer. The TPM also generates two new even rolling nonces Ne_1 and Ne_2 to continue the session with fresh nonces. These nonces will replace the nonces ne_1 and ne_2 that have been used in this command.

The TPM is intended to ensure the secrecy of the keys that are stored in its shielded memory. However, it seems clear that secrecy of these keys is not its only intended purpose. Indeed, the authorisation session mechanism and the HMACs provided for instance in input of each command would not be necessary to achieve this goal.

3.2.2 Modelling the TPM

In this section, we explain how the TPM commands and especially the mechanisms that are specific to security APIs can be formalised in ProVerif.

Non-monotonic global state. As explained in Section 3.1.2, one of the difficulties in reasoning about security APIs is the need of a non-monotonic global state. In order to be able to analyse the TPM with the ProVerif tool, we have to get rid of this non-monotonic global state. In the case of the TPM, we did that by introducing the assumption that only one command is executed in each authorisation session. Hence, rolling nonces that come from the authorisation session are now included in the local state. This assumption appears to be quite reasonable. Indeed, the TPM imposes the assumption itself whenever a command introduces a new authdata. Moreover, tools like TPM/J that provide software-level APIs also satisfy this assumption. Again to avoid the need of non-monotonic global state, 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.

Key table. In case of the TPM application, 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 need to 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 false attacks.

TPM commands. In our modelling we have two processes for each command: a user process and a TPM process. The user process models an honest user who makes a call

to the TPM while the TPM process 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.

Security properties. The TPM specification does not detail explicitly which security properties are intended to be guaranteed. First, it seems clear that the device is assumed to keep secret the keys that are stored in it. It is actually relatively easy to show that this secrecy property is indeed satisfied (at least for the fragment we considered in this work). However, the authorisation HMACs that accompany the commands and the HMACs provided by the TPM with its answer allow us to argue that the TPM certainly aims to also achieve the following properties:

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

In ProVerif, we model these properties as correspondence properties.

3.2.3 Some experiments

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 CertifyKey, CreateWrapKey, LoadKey2, and UnBind together. We consider the fixed version of each of these commands and we show in a last experiment 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 all our experiments, the security properties under test are the correspondence properties explained in the previous section. We described below the experiments done for the CertifyKey command together with some possible fixes[1].

Configuration 1. We consider a configuration with two honest keys loaded inside the TPM. ProVerif immediately discovers an attack 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 $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$. We patch the command CertifyKey by tagging

[1] All the files for these experiments are available at: `http://www.lsv.ens-cachan.fr/~delaune/TPM/`

the HMACs with two different tags, namely cfk_1 and cfk_2 instead of cfk. ProVerif is now able to verify the two correspondence properties.

Configuration 2. We add in the initial configuration another honest key sk_2' having the same authdata as a previous honest key already loaded in the TPM. ProVerif immediately discovers another attack. The attacker can exchange the key handle $\mathsf{handle}(auth_2, sk_2)$ with another handle having the same authdata. The TPM will answer by sending the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2'))$ whereas the user asked for the certificate $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$. This attack comes from the fact that the HMAC is only linked to the key via the authdata. 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 the 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. Considering this last fix, ProVerif is able to verify the two correspondence properties.

Configuration 3. We now assume that the attacker has his own key sk_i loaded onto the device meaning that he knows the authdata $auth_i$ associated to this key and also its public part $\mathsf{pk}(sk_i)$. Note that the attacker does not know sk_i that is stored inside the TPM (this key has been generated directly by the TPM). We immediately rediscover the attack of [GRS+07] (see Figure 3.1), 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 $\mathsf{cert}(sk_1, \mathsf{pk}(sk_2))$. ProVerif succeeds in proving the other correspondence property.

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

Trace: The attacker Charlie replaces the key to be certified by his own key.

USER	$\rightarrow$	TPM	:	request to open two authorisation 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, $\mathsf{hmac}(auth_1, \langle \mathsf{cfk}_1, \mathsf{pk}(sk_1), n, ne_1, no_1 \rangle)$, $\mathsf{handle}(auth_1, sk_1)$, $\mathsf{hmac}(auth_2, \langle \mathsf{cfk}_2, \mathsf{pk}(sk_2), n, ne_2, no_2 \rangle)$, $\mathsf{handle}(auth_2, sk_2)$
Charlie	$\rightarrow$	TPM	:	n, no_1, no_2, $\mathsf{hmac}(auth_1, \langle \mathsf{cfk}_1, \mathsf{pk}(sk_1), n, ne_1, no_1 \rangle)$, $\mathsf{handle}(auth_1, sk_1)$, $\mathsf{hmac}(auth_i, \langle \mathsf{cfk}_2, \mathsf{pk}(sk_i), n, ne_2, no_2 \rangle)$, $\mathsf{handle}(auth_i, sk_i)$
TPM	$\rightarrow$	Charlie	:	ne_1', ne_2', $\mathsf{cert}(sk_1, \mathsf{pk}(sk_i))$, ...

Figure 3.1 - Attack trace for the CertifyKey command (Configuration 3)

The attack of [GRS+07] comes from the fact that the attacker can replace the user's HMAC with one of his own. 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, we add a digest of each public

key inside each HMAC. For instance, the first HMAC built by the user will be now of the form: $\mathsf{hmac}(auth_1, \langle \mathsf{cfk1}, \mathsf{pk}(sk_1), \mathsf{pk}(sk_2), n, ne_1, no_1 \rangle)$. Now, 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.

Configuration 4. Lastly, we consider the fixed version of the commands CertifyKey, CreateWrapKey, LoadKey2, and UnBind (around 300 lines of ProVerif code). We consider a scenario where an honest key and a dishonest key are already loaded inside the TPM. Note that by using CreateWrapKey and 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. ProVerif is able to establish the 8 correspondence security properties. However, in one case, it is not able to verify the injective version of the property. This is due to a limitation of the tool and does not correspond to a real attack.

3.3 Reduction result for well-moded APIs

This formal work on well-moded APIs was primarily motivated by the example of RSA PKCS#11, which is widely deployed in industry. PKCS#11 is an interface between applications and cryptographic devices such as smartcards, Hardware Security Modules (HSMs), and USB key tokens. This standard is described in a large and complex specification [RSA04] (392 pages). Authentication of the user is by means of a PIN. However, if malicious code is running on the host machine, then the user PIN may easily be intercepted, *e.g.* by a tampered device driver, allowing an attacker to create his own sessions with the device. The PKCS#11 standard (see [RSA04, p. 31]) states that this kind of attack cannot however "compromise keys marked 'sensitive', since a key that is sensitive will always remain sensitive". The API as defined in the standard gives rise to a number of serious security vulnerabilities (see [Clu03a]). In practise, 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. The standard itself gives no advice on the subject, perhaps because to give incomplete advice might lead designers into a false sense of security.

To analyse this APIs, a non-monotonic global state is really needed. In particular, a crucial feature is to model the attributes associated to a key, and we need to model the fact that some attributes may be set or unset, and also the conditions under which these rules can be applied. A monotonic model of state would allow an attacker to take two mutually exclusive steps from the same state, permitting a lot of false attacks.

When we started this work, some efforts had already been made to formally analyse configurations of PKCS#11 considering however a global monotonic state. For instance, P. Youn [You04] used the first-order theorem prover Otter, and included in his model only the commands needed for Clulow's first attack. This model had one handle for each key, preventing the discovery of some attacks, and a monotonic model of state permitting false attacks. Tsalapati [Tsa07] used the AVISPA protocol analysis tools, included all the key management commands, but also used a monotonic model of state, and one handle for each key. She rediscovered a number of Clulow's attacks, but the limitations of the model prevented the discovery of the attacks we exhibit here.

3.3.1 The setting

In order to be able to model attributes of a key, we consider a finite set $\mathcal{A}$ of unary function symbols, disjoint from the signature $\mathcal{F}$ used to model messages. An *attribute term* is a term of the form $a(t)$ with $a \in \mathcal{A}$ and where t is a term built on the signature $\mathcal{F}$. Attribute terms will be interpreted as propositions. A *literal* is an expression of the form $a(t)$ or $\neg a(t)$.

Description language. We consider a rule-based description language close to a guarded command language *à la Dijkstra* (see [Dij75]) and to the multiset 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 formalised by the attributes that may be set or unset. The description of a system is given as a finite set of rules of the form:

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

where T and T' are sets of terms, L and L' are sets of literals and $\tilde{n}$ is a set of names. The formal semantics can be defined in a classical way as a transition system. Formally, a *state* is a pair (S, V) where S is a finite set of ground terms that represents the knowledge of the attacker, and V is a partial function from attribute terms to $\{\bot, \top\}$. Intuitively, the rule $T; L \xrightarrow{\mathsf{new}\,\tilde{n}} T'; L'$ can be fired from the state (S, V) if all terms in T are in S and if all the literals in L are evaluated to true by V. The effect of the rule is that terms in T' are added to S 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.

Security properties are expressed by the means of *queries*. 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 and a reachable state S that contains all the terms in $T\theta$ and such that all literals in $L\theta$ are evaluated to true.

Modelling PKCS#11. Here, we illustrate our description language on the key management operations of the API. We consider symmetric encryption only but the analysis of a more substantial fragment can be found in [26, 4]. We consider the signature $\Sigma_{\mathsf{enc}} = \{\mathsf{senc}, \mathsf{h}\}$. The symbol h of arity 2 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 model the attributes that are associated to handles by the means of the set $\mathcal{A}$. For the sake of simplicity, our running example only considers the attributes $\mathsf{extract}$, wrap, unwrap, $\mathsf{encrypt}$, $\mathsf{decrypt}$, and $\mathsf{sensitive}$.

The rules given in Figure 3.2 model a part of PKCS#11. We detail the first rule which allows wrapping of a symmetric key with a symmetric key. Intuitively the rule can be read as follows: if the attacker knows the handle $\mathsf{h}(x_1, y_1)$, *i.e.* a reference to a symmetric key y_1, and a second handle $\mathsf{h}(x_2, y_2)$, *i.e.* a reference to a symmetric key y_2, and if the attribute wrap is set for the handle $\mathsf{h}(x_1, y_1)$ (note that the handle is uniquely identified by the nonce x_1) and the attribute $\mathsf{extract}$ is set for the handle $\mathsf{h}(x_2, y_2)$ then the attacker may learn the wrapping $\mathsf{senc}(y_2, y_1)$, *i.e.* the encryption of y_2 with y_1.

Wrap :	$\mathsf{h}(x_1, y_1), \mathsf{h}(x_2, y_2);\ \mathsf{wrap}(x_1), \mathsf{extract}(x_2)$	$\longrightarrow$	$\mathsf{senc}(y_2, y_1)$
Unwrap :	$\mathsf{h}(x_1, y_2), \mathsf{senc}(y_1, y_2);\ \mathsf{unwrap}(x_1)$	$\xrightarrow{\text{new } n_1}$	$\mathsf{h}(n_1, y_1);\ \mathsf{extract}(n_1),\ \mathsf{L}$
KeyGenerate :		$\xrightarrow{\text{new } n_1, k_1}$	$\mathsf{h}(n_1, k_1); \neg\mathsf{extract}(n_1),\ \mathsf{L}$
SEncrypt :	$\mathsf{h}(x_1, y_1), y_2;\ \mathsf{encrypt}(x_1)$	$\longrightarrow$	$\mathsf{senc}(y_2, y_1)$
SDecrypt :	$\mathsf{h}(x_1, y_1), \mathsf{senc}(y_2, y_1);\ \mathsf{decrypt}(x_1)$	$\longrightarrow$	y_2
Set_Wrap :	$\mathsf{h}(x_1, y_1);\ \neg\mathsf{wrap}(x_1)$	$\longrightarrow$	$\mathsf{wrap}(x_1)$
UnSet_Wrap :	$\mathsf{h}(x_1, y_1);\ \mathsf{wrap}(\mathsf{x}_1)$	$\longrightarrow$	$\neg\mathsf{wrap}(x_1)$
Set_Encrypt :	$\mathsf{h}(x_1, y_1);\ \neg\mathsf{encrypt}(x_1)$	$\longrightarrow$	$\mathsf{encrypt}(x_1)$
UnSet_Encrypt :	$\mathsf{h}(x_1, y_1);\ \mathsf{encrypt}(x_1)$	$\longrightarrow$	$\neg\mathsf{encrypt}(x_1)$
⋮		⋮	

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 3.2 - PKCS#11 key management subset (symmetric encryption only).

3.3.2 Reduction result and decidability

In this section, we establish a result that allows us to reduce the search space when we are looking for an attack, *i.e.* when we try to decide the satisfiability of a query. This result is quite general and can be applied as soon as the rules are *well-moded*. 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 emphasise that we do not have a typing assumption. We show that if there is an attack, then there is an attack where bitstrings are used for the purpose they were originally created. Unfortunately, deciding whether a query is satisfiable from an initial state is still undecidable in this setting. We show how to get decidability by considering an additional assumption.

Well moded rules. We consider a set of modes Mode and we assume that there exists a *mode* function such that $\mathsf{M}(\mathsf{f}, i)$ is defined for every symbol f and every integer i such that $1 \leq i \leq ar(\mathsf{f})$. We also assume that a function sig returns the mode to which a symbol f belongs. We extend the function sig to terms by considering its root symbol. Intuitively, a position of a term t is *well-moded* if the subterm at that position is of the expected mode *w.r.t.* to the function symbol immediately above it. A term is well-moded if all its positions are well-moded. This notion is extended as expected to literals, sets of terms, rules, queries, and states.

Note that any term can be seen as a well-moded term if there is a unique mode, *e.g.* Msg. However, such a result will not be really useful to reduce the search space. The rules described in Figure 3.2 and the rules needed to represent the deduction capabilities of the attacker are well-moded *w.r.t.* the modes and signature function described below. This set of modes

allows us to bound the length of a well-moded term.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{h} & : & \mathsf{Nonce} \times \mathsf{Key} \to \mathsf{Handle} & x_1, x_2, n_1 & : & \mathsf{Nonce} \\ \mathsf{senc} & : & \mathsf{Key} \times \mathsf{Key} \to \mathsf{Cipher} & y_1, y_2, k_1 & : & \mathsf{Key} \end{array}$$

$$\mathsf{att} \ : \ \mathsf{Nonce} \to \mathsf{Attribute} \quad \text{for all } \mathsf{att} \in \mathcal{A}$$

Decidability. First, we show that in a system induced by well-moded rules, only well-moded terms need to be considered when checking for the satisfiability of a well-moded query. More formally, we have that:

Theorem 3.1 *[26, 4] Let $\mathcal{R}$ be a set of well-moded rules. Let (S_0, V_0) be a well-moded state (the initial state) such that for any possible 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 derivation witnessing this fact that only involves well-moded states.*

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 well-moded. Note also that the modes used in our modelling of PKCS# 11 imply that all well-moded terms have bounded message length. This is not sufficient for decidability. Indeed, undecidability proofs [DLM04, TEB05] for security protocols with bounded message length and unbounded number of nonces can be easily adapted to our setting. Therefore, in order to get a decidability result, we bound the number of atomic data of each mode.

3.3.3 Analysing PKCS#11 and some proprietary extensions

The decision procedure arising from Theorem 3.1 for a bounded number of keys and handles has been implemented. Actually, the propositional encoding is generated in a syntax suitable for the model checker NuSMV [CCG+02]. We then asked NuSMV to check whether the security property (secrecy of sensitive keys) holds.

Methodology. PKCS#11 is a standard designed to promote interoperability, not a tighthly 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 with the secrecy of sensitive keys as security property (see PKCS#11 manual [RSA04, p. 31]). First, we perform experiments involving symmetric keys only, and try to restrict the API until a secure configuration was found. For this, as suggested by J. Clulow [Clu03a], we declare some conflicting attributes which means that some appropriate conditions are added to the left-hand side of the Set rules. We also study the notion of sticky attributes, *i.e.* those that, once set, cannot be unset, and those which once unset, cannot be set anymore. This way of restricting the API was actually suggested by the PKCS standard.

We then added asymmetric keys, and repeated the process. A secure configuration is obtain by adding the 'trusted keys' mechanism introduced in PKCS#11 v2.20. We also carried out some experiments modelling the algebraic properties of the ECB mode of encryption. Finally, we considered the proprietary extensions used by two commercial providers of cryptographic hardware.

Summary of our experiments. All the files for our experiments are available at:

`http://www.lsv.ens-cachan.fr/~steel/pkcs11/`

In our first set of experiments (see [26]), we examine key separation attacks, similar to those found by J. Clulow [Clu03a]. We constrain our PKCS#11 API by repeatedly banning certain combinations of key attributes, each time obtaining a new (longer) attack. Finally we investigate some solutions involving for example the 'trusted keys' mechanism.

Concerning the two proprietary extensions (see [4]), we find that both permitted secure configurations to be found, but both require careful attention to other details in order to avoid vulnerabilities. However, we have only examined small bounded models.

3.3.4 Related results

Our result has been extended by S. Fröschle and G. Steel to deal with an unbounded number of fresh data [FS09]. To achieve this, they consider APIs having a particular policy. Actually, APIs outside this class are likely to be problematic. Thus, this restriction appears to be quite natural. Then, they show that these APIs can be safely abstracted to APIs where the attributes are fixed. It is thus sufficient to consider a small bounded model. A positive result will guarantee security in the unbounded case. However, their abstract model may suggest false attacks.

In our work, we showed how difficult it is to prevent attacks: the commands can be restricted to prevent certain conflicting attributes from being set on the same object, but still more attacks arise. However, we do not check whether any real devices following the standard actually implement key management like this, since much of the functionality is optional. In [BCFS10], M. Bortolozzo *et al*, motivated by our work, developed a tool, called Tookan. Their tool reverse-engineers the particular token in use to deduce its functionality, constructs a model of its API (following the model presented in Section 3.3.1), and then executes any attack trace found by the model checker directly on the tokens. They use the SATMC model checker relying on our result (see Theorem 3.1) to reduce the search space. They have tested the Tookan tool on 17 commercially available tokens: 9 were vulnerable to attacks, while the other 8 had severely restricted functionality.

3.4 A formal theory of key conjuring

In this section, we describe a formalism for *key conjuring*, the process by which an attacker obtains an unknown, encrypted key by repeatedly calling a cryptographic API function with random values in place of keys. We propose a formalism for detecting computationally feasible key conjuring operations, incorporated into a Dolev-Yao style model of the security API. Then, we show that security in the presence of key conjuring operations is decidable for a particular class of APIs, which includes the key management API of IBM's Common Cryptographic Architecture (CCA). In particular, this requires consideration of the algebraic properties of the exclusive-or operation.

Formal work on the CCA first concentrated on rediscovering the attacks on the original version of the API [Ste05, YAB+05], and then on proving both Bond's proposed fixes [CM06], and the fixes IBM actually implemented [CKS07], to be secure. However, these works made an informal approximation of the ability of the intruder to 'conjure' keys, a trick used several

times in Bond's attacks. To explain precisely what key conjuring is, we first need to introduce some notation.

3.4.1 Some definitions

Before introducing key conjuring, we first define the class of APIs considered in this section.

Modelling. We consider a set of modes. Terms which respect these modes are said to be *well-moded*. It includes a set of base mode Base and a set of ciphertext mode Cipher. Moreover we assume that our function symbols have the following mode:

$$\begin{array}{rcccccc} \oplus & : & \mathsf{Base} & \times & \mathsf{Base} & \to & \mathsf{Base} \\ \mathsf{senc} & : & \mathsf{Base} & \times & \mathsf{Base} & \to & \mathsf{Cipher} \\ \mathsf{sdec} & : & \mathsf{Cipher} & \times & \mathsf{Base} & \to & \mathsf{Base} \end{array}$$

We equip the signature with an equational theory $\mathsf{E_{API}}$ that models the algebraic properties of our operators:

$$\mathsf{E_{API}} \stackrel{\mathsf{def}}{=} \left\{ \begin{array}{llll} \mathsf{senc}(\mathsf{sdec}(x,y),y) = x & x \oplus 0 = x & x \oplus (y \oplus z) = (x \oplus y) \oplus z \\ \mathsf{sdec}(\mathsf{senc}(x,y),y) = x & x \oplus x = 0 & x \oplus y = y \oplus x \end{array} \right.$$

In the CCA API, as in many others, symmetric keys are subject to parity checking. The 4758 uses the DES (and 3DES) algorithm for symmetric key encryption. A (single length) DES key consists of 64 bits in total, which is divided into eight groups, each consisting of seven key bits and one associated parity bit. For a key of odd parity, each parity bit must be set so that the overall parity of its group is odd. For a key of even parity, the parity bits must be set so that all groups are of even parity. If the groups have mixed parities, then the key is of undefined parity and considered invalid. These parity considerations are important for our analysis of key conjuring, and are represented in our formalism by occurrences of the predicate symbols $\mathsf{chkEven}$ and chkOdd, each having a term as argument. Moreover, we have some deduction rules to infer parity from known facts.

Intruder capabilities and the API behaviour are described using *API rules*.

Definition 3.1 *An* API rule *is a rule of the form* $\mathsf{chk}_1(u_1), \ldots, \mathsf{chk}_k(u_k), x_1, \ldots, x_n \to t$ *where* $x_1, \ldots, x_n$ *are variables,* t *is a term such that* $var(t) \subseteq \{x_1, \ldots, x_n\}$, $u_1, \ldots, u_k$ *are terms of* Base *mode not headed with* $\oplus$, $\mathsf{chk}_i \in \{\mathsf{chkOdd}, \mathsf{chkEven}\}$, $1 \leq i \leq k$. *We also assume that the rule only involves well-moded terms with at most one encryption symbol that has to occur at their root position.*

This class allows us in particular to model the attacker capabilities in a Dolev-Yao style and the symmetric key management subset of the IBM 4758 API [CCA06].

Key conjuring. As we have seen, key management APIs like the CCA keep working keys outside the HSM, safely encrypted, so that they can only be used by sending them back into the HSM. What happens when an intruder wants to use a particular command in an attack, but does not have access to an appropriate key? For example, suppose he has no data keys

(terms of the form $\mathsf{senc}(\mathsf{d1}, \mathsf{km} \oplus \mathsf{data})$), but wants to use the ENCRYPT DATA command. In an implicit decryption formalism, the command is defined like this:

$$\text{ENCRYPT DATA} \quad x,\ \mathsf{senc}(xkey, \mathsf{km} \oplus \mathsf{data}) \ \rightarrow\ \mathsf{senc}(x, xkey).$$

This suggests that the command cannot be used if the intruder does not have a data key. However, in reality, an intruder could just guess a 64 bit value and use that in place of the data key. The HSM will decrypt the guessed value under $\mathsf{km} \oplus \mathsf{data}$, and check the parity of the resulting 64 bit term to see if it is a valid key before encrypting the data. Usually, the check will fail and the HSM will refuse to process the command, but if the intruder guesses randomly, he can expect that 1 in every 256 guessed values will result in a valid key. This notion is captured by our formalism, in which we write the ENCRYPT DATA command like this:

$$\text{ENCRYPT DATA} \quad \mathsf{chkOdd}(\mathsf{sdec}(y, \mathsf{km} \oplus \mathsf{data})),\ x,\ y\ \rightarrow\ \mathsf{senc}(x, \mathsf{sdec}(y, \mathsf{km} \oplus \mathsf{data}))$$

It may seem useless for the intruder to simply guess values, since the result is a term he knows encrypted under an unknown key, but used cleverly, this technique can result in serious attacks. For example, Bond's so called Import/Export attack [Bon01], uses key conjuring to convert a PIN derivation key into an encryption key, allowing an intruder to generate the PIN for any given account number. Further attacks using key conjuring had been discovered by then, [CB03, Clu03a], on both the CCA API and other APIs.

Note that a straightforward 'explicit decryption' model is not sufficient for a key conjuring analysis, since though this allows an attack like Bond's be discovered, it doesn't take into account parity checks. This means that the model cannot distinguish between feasible and non-feasible key conjuring steps, leading to false attacks. For example, an explicit decryption model without parity checking would allow an intruder to conjure several keys at the same time which in practise is highly unlikely. The transformation we proposed to model key conjuring ensures that the intruder has to guess values for at most one parity check. Moreover, this transformation is generic enough to deal with any API made up of rules satisfying the conditions given in Definition 3.1. Given an API rule R, our transformation will return a set of rules, denoted by $\mathsf{KeyCj}(\mathsf{R})$.

Example 3.1 *The key conjuring rule obtained with our transformation from the rule* ENCRYPT DATA *is as follows:*

$$\text{ENCRYPT DATA} \quad x\ \stackrel{\mathsf{new}\, n}{\rightarrow}\ \mathsf{senc}(x, \mathsf{sdec}(n, \mathsf{km} \oplus \mathsf{data})),\ n,\ \mathsf{chkOdd}(\mathsf{sdec}(n, \mathsf{km} \oplus \mathsf{data}))$$

To encrypt a data x with a key $xkey$, it is necessary to provide to the interface the data and the ciphertext $\mathsf{senc}(xkey, \mathsf{km} \oplus \mathsf{data})$. *However, the intruder can simply send a random sequence n instead of this ciphertext. If it happens that* $\mathsf{sdec}(n, \mathsf{km} \oplus \mathsf{data})$ *satisfies the parity condition then the rule will return* $\mathsf{senc}(x, k)$ *where* $k = \mathsf{sdec}(n, \mathsf{km} \oplus \mathsf{data})$.

3.4.2 Decidability result

We consider here the problem of deciding whether a particular term, for example a PIN derivation key, can be learnt by an attacker. The intruder starts with a fixed set of terms that constitute his *initial knowledge*. He can then use the rules of the API and also the key conjuring variants of the rules in any order to extend his knowledge. Each time the

adversary wants to conjure a key, it requires a significant amount of access to the API. We assume in what follows that the use of these rules by the attacker is limited. This is modelled by introducing a parameter k that bounds the maximum number of applications of the key conjuring rules. The value of k could be set based on the amount of time an attacker may have access to a live HSM, based on physical security measures, auditing procedures in place, ...

We write $S \vdash^{\mathcal{A}_2 \leq k}_{\mathcal{A}_1, \mathsf{E}_{\mathsf{API}}} u$ if u is deducible from S by using the rules in $\mathcal{A}_1$ and at most k instances of the rule in $\mathcal{A}_2$ (modulo the theory $\mathsf{E}_{\mathsf{API}}$). The *security problem* we are interested in can be defined as follows:

Entries: A finite set $\mathcal{A}$ of API rules, a set S of ground facts that is consistent (the initial knowledge of the attacker), a ground term s (the secret) and a bound $k \in \mathbb{N}$ (number of key conjuring steps).

Question: Is the secret s deducible from S by using the rules in $\mathcal{A}$ and at most k instances of rules in $\mathsf{KeyCj}(\mathcal{A})$ (modulo $\mathsf{E}_{\mathsf{API}}$), *i.e.* does $S \vdash^{\mathsf{KeyCj}(\mathcal{A}) \leq k}_{\mathcal{A}, \mathsf{E}_{\mathsf{API}}} s$?

Theorem 3.2 *[33] The security problem defined above is decidable for well-formed APIs.*

The notion of well-formed API is formally defined in [33]. It mostly restricts the APIs by imposing some natural conditions that are in particular satisfied by the IBM 4758 API. For instance, it imposes that the API only checks the parity of objects that have to be used in generating the output. This is quite natural since we would not expect an API to check the parity of a term that is subsequently discarded.

Related work. Our class of API rules is related to the class proposed in [CKS07]. It is shown that secrecy preservation is decidable for an unbounded number of sessions for API rules with XOR, provided that these rules satisfy a condition closed to our notion of *well-moded terms*. However, there are two main differences between the class of well-formed API rules studied in this chapter and the class defined in [CKS07]. First, we consider here an equational theory with explicit decryption. This is necessary for modelling key conjuring. Second, because of the use of explicit destructor symbol, we have to deal with an infinite number of well-moded terms.

To the best of our knowledge, there exist only two other decidable classes [CLC03, VSS05] for secrecy preservation for protocols with XOR, for an unbounded number of sessions. In both cases, the main difference with our class is that we make restrictions on the combination of functional symbols rather than on the occurrences of variables. As a consequence, our class is incomparable to the two existing ones. In particular, the IBM CCA protocol cannot be modelled in either of these two other classes.

More recently, R. Küsters and T. Truderung have designed and implemented an algorithm for checking trace properties on protocols using the XOR operator [KT11]. This tool takes as input Horn clauses with XOR (modelling the protocols and the security property) and translates them into Horn clauses (without XOR) that can be handled by ProVerif. In particular, using their implementation (together with ProVerif), they found a new attack. They do not consider explicit destructors, and thus they do not take into account key conjuring.

3.5 Perspectives

In this chapter, we have analysed three different APIs and we have presented two decision procedures. Despite the undecidability results for verifying protocols in presence of an unbounded number of sessions, we have shown that it is possible to exploit the specificities of the APIs to get decision procedures without limiting the number of sessions. The abstraction and over-approximation of protocols by a set of Horn clauses (the method used in the ProVerif tool) is a very successful method in practice to analyse cryptographic protocols. Using this tool, we have also performed an interesting analysis of the TPM.

The results presented in this chapter can be extended in several ways. For instance, concerning our work on the TPM, we foresee extending our model with more commands such as those involved in key migration. We also plan to model the TPM's *platform configuration registers* (PCRs) which allow one to condition some commands on the current value of a register. PCRs are crucial when using the TPM for checking the integrity of a system. Modelling the PCRs and the commands for manipulating these registers for automated verification seems to be a challenging task, because of the need for a non-monotonic global state.

Regarding our work on key conjuring, it seems natural to extend our decision procedure to a larger class of APIs, incorporating pairing and further cryptographic primitives. Even if key conjuring allows one to capture a large class of brute-force attacks, there still remain some feasible brute-force attacks that are not captured by our model, *e.g.* parallel-key search attacks, meet-in-the-middle attack, ... The meet-in-the-middle attack is a cryptographic attack which, like the birthday attack, makes use of a space-time tradeoff. While the birthday attack attempts to find two values in the domain of a function that map to the same value in its range, the meet-in-the-middle attack attempts to find a value in each of the range and domain of the composition of two functions such that the forward mapping of one through the first function is the same as the inverse image of the other through the second function – quite literally meeting in the middle of the composed function. However, in order to capture feasible brute-force attacks only, it seems important to introduce a quantitative model. Some efforts have already been done in this direction (*e.g.* [AMRV06]), this is however insufficient for modelling the attacks mentioned here (*e.g.* [SC07]).

More generally, it seems important to propose a unified framework allowing us to model a large number of security APIs. This seems quite challenging since even if security APIs have some common features (*e.g.* key table, ...) we often need to model in an accurate way some of their specific features to conduct a reasonable analysis. An important challenge is to develop decision procedures allowing one to take into account non-monotonic global state. This is needed for the analysis of most of the APIs, *e.g.* the key attributes in PKCS #11, the PCR used in the TPM application, ... We have seen that the method based on over-approximation such as the use of Horn clauses is very successful in practice; it has however limitations for protocols that are based on databases of keys such as security APIs. Recently, some advances have been made to extend the scope of these over-approximation methods by defining a new way of abstraction that can handle such databases [Möd10]. In order to obtain efficient tools, the specificities of the security APIs could be helpful. This will probably lead to an automatic verifier tailored for this specific application.

Part II

Privacy-type security properties

Chapter 4

Modelling in applied pi calculus

Contents

Formal methods have proved their usefulness for precisely analysing the security of protocols. However, most existing results focus on trace properties, that is, statements that something bad never occurs on any execution trace of a protocol. Secrecy and authentication, and more generally all the properties studied in Chapter 2 and Chapter 3, are typical examples of trace properties. There are however several security properties, which cannot be defined (or cannot be naturally defined) as trace properties and require the notion of *observational equivalence*. Intuitively, two processes P and Q are equivalent, denoted by $P \approx Q$, if for any process O (the observer) the processes $P \mid O$ and $Q \mid O$ are equally able to emit on a given channel. This means that the process O cannot observe any difference between the processes P and Q.

We focus here on the definition proposed in the context of applied pi calculus [AF01], which is well-suited for the analysis of security protocols. First, we present the applied pi calculus in Section 4.1. Then, in Section 4.2, we formally define the notions of observational

and trace equivalence, and we study the relationship between these two notions. Observational equivalence is crucial when specifying properties like anonymity that states that an observer cannot distinguish the case where A is talking from the case where B is talking. We have shown that privacy related properties involved in electronic voting protocols [6] or in the context of vehicular ad hoc networks [15] also rely on equivalence as a key notion. Those applications are detailed in Section 4.3. Lastly, we give a brief description of the ProVerif tool [Bla01]. This is a well-established protocol verifier able to check observational equivalence. However, the method used by ProVerif is not complete and the tool is actually unable to prove the equivalences presented in Section 4.3.

4.1 Applied pi calculus

The *applied pi calculus* [AF01] is a derivative of the pi calculus that is specialised for modelling cryptographic protocols. Participants in a protocol are modelled as processes, and the communication between them is modelled by means of message passing.

4.1.1 Syntax

We consider a set of *names*, which is split into the set $\mathcal{N} = \{a, b, k, n, \ldots\}$ of names of *base types* and the set $\mathcal{C}h$ of names of *channel type* (which are used to name communication channels). We also consider a set of *variables* $\mathcal{X} = \{x, y, \ldots\}$, and a *signature* $\mathcal{F}$ consisting of a finite set of *function symbols*. We rely on a sort system for terms. The details of the sort system are unimportant, as long as it distinguishes *base types* from the *channel type*. We suppose that function symbols only operate on and return terms of base types. *Terms* are defined as names, variables, and function symbols applied to other terms. We denote by $\mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X})$ the set of terms built on $\mathcal{F}$ and $\mathcal{N} \cup \mathcal{X}$. Of course function symbol application must respect sorts and arities.

Example 4.1 *Let* $\mathcal{F} = \{\mathsf{aenc}/2,\ \mathsf{adec}/2,\ \mathsf{pk}/1,\ \langle\ \rangle/2,\ \mathsf{proj}_1/1,\ \mathsf{proj}_2/1\}$ *be a signature containing function symbols for asymmetric encryption, decryption and pairing, each of arity* 2*, as well as projection symbols and the function symbol* pk*, each of arity* 1*. The term* $\mathsf{pk}(sk)$ *represents the public counterpart of the private key* sk*.*

In the applied pi calculus, one has *plain processes*, denoted P, Q, R and *extended processes*, denoted by A, B, C. Plain processes are built up in a similar way to processes in pi calculus except that messages can contain terms rather than just names. Extended processes add *active substitutions* and restriction on variables. In the grammar described 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.

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

$\{^M/_x\}$ is the active substitution that replaces the variable x with the term M. Active substitutions generalise the "let" construct: $\mathtt{new}\ x.(\{^M/_x\} \mid P)$ corresponds exactly to "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 say that an extended process is *closed* if all its variables are either bound or defined by an active substitution. An *evaluation context* $C[\_]$ is an extended process with a hole instead of an extended process.

Active substitutions are useful because they allow us to map an extended process A to its *frame*, denoted $\phi(A)$, by replacing every plain process in A with 0. A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ accounts for the set of terms statically possessed by the intruder (but does not take into account for A's dynamic behaviour). The *domain* of a frame φ, denoted by $dom(\varphi)$, is the set of variables for which φ defines a substitution (those variables x for which φ contains a substitution $\{^M/_x\}$ not under a restriction on x).

Example 4.2 *Consider the signature $\mathcal{F}$ of Example 4.1. Let A be the following process made up of three components in parallel:*

$$A = \mathit{new}\, s, sk, x_1.(\mathit{out}(c_1, x_1) \mid \mathit{in}(c_1, y).\mathit{out}(c_2, \mathsf{adec}(y, sk)) \mid \{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}).$$

The first component publishes the message $\mathsf{aenc}(s, \mathsf{pk}(sk))$ stored in x_1 by sending it on c_1. The second receives a message on c_1, uses the secret key sk to decrypt it, and forwards the result on c_2. We have $\phi(A) = \mathit{new}\, s, sk, x_1.\{^{\mathsf{aenc}(s,\mathsf{pk}(sk))}/_{x_1}\}$ and $dom(\phi(A)) = \emptyset$ (since x_1 is under a restriction).

4.1.2 Semantics

We briefly recall the operational semantics of the applied pi calculus (see [AF01] for details). First, we associate an *equational theory* E to the signature $\mathcal{F}$. The equational theory is defined by a set of equations of the form $M = N$ where $M, N \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and induces an equivalence relation over terms: $=_{\mathsf{E}}$ is the smallest congruence relation on terms, which contains all equations $M = N$ in E, and that is closed under substitution of terms for variables.

Example 4.3 *Considering the signature $\mathcal{F}$ of Example 4.1 we define the equational theory $\mathsf{E}_{\mathsf{aenc}}$ by the equations $\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$ and $\mathsf{proj}_i(\langle x_1, x_2\rangle) = x_i$ for $i \in \{1, 2\}$.*

Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, by application of evaluation contexts, and satisfying some further basic structural rules such as $A \mid 0 \equiv A$, associativity and commutativity of $\mid$, binding-operator-like behaviour of $\mathtt{new}$, and when $M =_{\mathsf{E}} N$ the equivalences

$$\mathtt{new}\ x.\{^M/_x\} \equiv 0 \qquad \{^M/_x\} \equiv \{^N/_x\} \qquad \{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$$

Example 4.4 *Let $P = \mathit{new}\ s, sk.(\mathit{out}(c_1, \mathsf{aenc}(s, \mathsf{pk}(sk))) \mid \mathit{in}(c_1, y).\mathit{out}(c_2, \mathsf{adec}(y, sk)))$. The process P is structurally equivalent to the process A given in Example 4.2. We have also that $\phi(P) = 0 \equiv \phi(A)$.*

The operational semantics of processes in the applied pi calculus is defined by rules defining two relations: *structural equivalence* (described above) and *internal reduction*, noted $\xrightarrow{\tau}$. Internal reduction $\xrightarrow{\tau}$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that:

$$\mathtt{out}(a,x).P \mid \mathtt{in}(a,x).Q \xrightarrow{\tau} P \mid Q \qquad \mathtt{if}\ M = M\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau} P$$

$$\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q \xrightarrow{\tau} Q \text{ where } M, N \text{ are ground terms such that } M \neq_{\mathtt{E}} N$$

The operational semantics is extended by a *labelled* operational semantics enabling us to reason about processes that interact with their environment. Labelled operational semantics defines the relation $\xrightarrow{\ell}$ where ℓ is either an input or an output. We adopt the following rules in addition to the internal reduction rules. Below, a is a channel name, u is a metavariable, and x is a variable of base type.

In $\quad \mathtt{in}(a,x).P \xrightarrow{in(a,M)} P\{^{M}/_{x}\}$

Out-Atom $\quad \mathtt{out}(a,u).P \xrightarrow{out(a,u)} P$

Open-Atom $\quad \dfrac{A \xrightarrow{out(a,u)} A' \qquad u \neq a}{\mathtt{new}\ u.A \xrightarrow{\mathtt{new}\ u.out(a,u)} A'}$

Scope $\quad \dfrac{A \xrightarrow{\ell} A' \quad u \text{ does not occur in } \ell}{\mathtt{new}\ u.A \xrightarrow{\ell} \mathtt{new}\ u.A'}$

Par $\quad \dfrac{A \xrightarrow{\ell} A' \quad \begin{array}{r} bn(\ell) \cap fn(B) = \emptyset \\ bv(\ell) \cap fv(B) = \emptyset \end{array}}{A \mid B \xrightarrow{\ell} A' \mid B}$

Struct $\quad \dfrac{A \equiv B \quad B \xrightarrow{\ell} B' \quad A' \equiv B'}{A \xrightarrow{\ell} A'}$

Note that the labelled transition is not closed under application of evaluation contexts. Moreover the output of a term M needs to be made "by reference" using a restricted variable and an active substitution.

4.2 Observational equivalence versus trace equivalence

Let $\mathcal{A}$ be the alphabet of actions (in our case this alphabet is infinite) where the special symbol $\tau \in \mathcal{A}$ represents an unobservable action. For every $\alpha \in \mathcal{A}$ the relation $\xrightarrow{\alpha}$ has been defined in Section 4.1.2. We consider the relation $\overset{\alpha}{\mapsto}$ that is the restriction of $\xrightarrow{\alpha}$ on closed extended processes. For every $w \in \mathcal{A}^*$ the relation $\overset{w}{\mapsto}$ on closed extended processes is defined in the usual way. By convention $A \overset{\epsilon}{\mapsto} A$ where ϵ denotes the empty word. For every $s \in (\mathcal{A} \setminus \{\tau\})^*$, the relation $\overset{s}{\Mapsto}$ on extended processes is defined by: $A \overset{s}{\Mapsto} B$ if, and only if, there exists $w \in \mathcal{A}^*$ such that $A \overset{w}{\mapsto} B$ and s is obtained from w by erasing all occurrences of τ. Intuitively, $A \overset{s}{\Mapsto} B$ means that A transforms into B by experiment s.

4.2.1 Observational equivalence

Intuitively, two processes are *observationally equivalent* if they cannot be distinguished by any active attacker represented by any context. Observational equivalence can be used to formalise many interesting security properties, in particular privacy related properties (see Section 4.3). The universal quantification over contexts makes observational equivalence difficult to verify. Hence, an alternative characterization, namely *labelled bisimilarity*, is introduced in [AF01]. This characterization, recalled in Definition 4.2, relies on a direct comparison of labelled

transitions rather than on contexts. First, we introduce a notion of intruder's knowledge that has been extensively studied. Several results about this notion are presented in Chapter 5.

Definition 4.1 (static equivalence $\sim$) *Two terms M and N are* equal in the frame ϕ, *and we write $(M =_{\mathsf{E}} N)\phi$, if there exists $\tilde{n}$ and a substitution σ such that $\phi \equiv \mathtt{new}\,\tilde{n}.\sigma$, $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, and $M\sigma =_{\mathsf{E}} N\sigma$. Two closed frames ϕ_1 and ϕ_2 are* statically equivalent, *$\phi_1 \sim_{\mathsf{E}} \phi_2$ (or simply $\phi_1 \sim \phi_2$ when E is clear from the context), when*

- *$dom(\phi_1) = dom(\phi_2)$, and*
- *for all terms M, N we have that $(M =_{\mathsf{E}} N)\phi_1$ if and only if $(M =_{\mathsf{E}} N)\phi_2$.*

Example 4.5 *Consider the two frames $\phi_0 = \{^{\mathsf{aenc}(s_0,\mathsf{pk}(sk))}/_{x_1}\}$ and $\phi_1 = \{^{\mathsf{aenc}(s_1,\mathsf{pk}(sk))}/_{x_1}\}$. We have $(\mathsf{adec}(x_1, sk) =_{\mathsf{E}_{\mathsf{aenc}}} s_0)\phi_0$ whereas $(\mathsf{adec}(x_1, sk) \neq_{\mathsf{E}_{\mathsf{aenc}}} s_0)\phi_1$, thus $\phi_0 \not\sim \phi_1$. However, we have that $\mathtt{new}\, sk.\phi_0 \sim \mathtt{new}\, sk.\phi_1$. This is a non trivial equivalence. Intuitively, there is no test that allows one to distinguish the two frames since neither the decryption key, nor the encryption key are available.*

Definition 4.2 (labelled bisimilarity $\approx_\ell$) Labelled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\ \mathcal{R}\ B$ implies*

1. *$\phi(A) \sim \phi(B)$,*
2. *if $A \overset{\tau}{\mapsto} A'$, then $B \overset{\epsilon}{\Mapsto} B'$ and $A'\ \mathcal{R}\ B'$ for some B',*
3. *if $A \overset{\ell}{\mapsto} A'$ and $bn(\ell) \cap fn(B) = \emptyset$ then $B \overset{\ell}{\Mapsto} B'$ and $A'\ \mathcal{R}\ B'$ for some B'.*

Example 4.6 *Consider the theory $\mathsf{E}_{\mathsf{aenc}}$, the processes $P_0 = \mathtt{out}(c, \mathsf{aenc}(s_0, \mathsf{pk}(sk)))$ and $Q_0 = \mathtt{out}(c, \mathsf{aenc}(s_1, \mathsf{pk}(sk)))$. We have $\mathtt{new}\, sk.P_0 \approx_\ell \mathtt{new}\, sk.Q_0$ whereas $P_0 \not\approx_\ell Q_0$. These results are direct consequences of the static (in)equivalence relations stated in Example 4.5.*

4.2.2 Trace equivalence

For every closed extended process A, we define its set of traces, each trace consisting in a sequence of actions together with the sequence of sent messages

$$\mathsf{trace}(A) = \{(s, \phi(B)) \mid A \overset{s}{\Mapsto} B \text{ for some closed extended process } B\}.$$

Definition 4.3 (trace inclusion $\sqsubseteq_t$, trace equivalence $\approx_t$) *Let A and B be two closed extended processes, $A \sqsubseteq_t B$ if for every $(s, \varphi) \in \mathsf{trace}(A)$ such that $bn(s) \cap fn(B) = \emptyset$, there exists $(s', \varphi') \in \mathsf{trace}(B)$ such that $s = s'$ and $\varphi \sim \varphi'$. The processes A and B are* trace equivalent, *denoted by $A \approx_t B$, if $A \sqsubseteq_t B$ and $B \sqsubseteq_t A$.*

It is well-known that observational equivalence (or labelled bisimilarity) implies trace equivalence whereas the converse is false in general.

4.2.3 Determinacy

J. Engelfriet has shown that observational equivalence and trace equivalence coincide for a process algebra with atomic actions, when processes are *determinate* [Eng85].

Definition 4.4 (determinacy) *Let $\cong$ be an equivalence relation on closed extended processes. A closed extended process A is* $\cong$-determinate *if whenever $A \stackrel{s}{\Rightarrow} B$, $A \stackrel{s}{\Rightarrow} B'$ and $\phi(B) \sim \phi(B')$ then $B \cong B'$.*

Fixing the equivalence relation yields to potentially different notions of determinacy, *e.g. observation determinacy* (for $\cong \; := \; \approx$) and *trace determinacy* (for $\cong \; := \; \approx_t$). Following J. Engelfriet techniques, we have shown (see [21]) that these two notions of determinacy coincide. So we say that an extended process is *determinate* if it satisfies any of these two notions. Then, we have extended the result of J. Engelfriet [Eng85] to processes of the applied pi calculus, showing that observational equivalence and trace equivalence coincide when processes are determinate.

Theorem 4.1 *[21] Let A and B be two closed extended processes that are determinate.*

$$A \approx_t B \text{ implies } A \approx_\ell B.$$

Originally, the notion of observational equivalence (or equivalently labelled bisimilarity) has been introduced to approximate trace equivalence [AG99]. The fact that observational equivalence is based on a notion of step-by-step simulation between processes makes this notion sometimes easier to prove directly. Even if the notion of observational equivalence is quite strong, it is also used to express privacy type properties. In the next section, we use $\approx$ to represent $\approx_t$ or $\approx_\ell$ (leading of course to different notions of equivalence when the underlying processes are not determinate).

4.3 Applications

In this section, we illustrate how security properties can be expressed by the means of equivalences. We detailed three applications. Actually, equivalences are also useful to model many other security properties, *e.g.* anonymity [AF04], untraceability [ACRR10, BCdH10], ...

4.3.1 Guessing attacks on password-based protocols

Guessing attacks are a kind of dictionary attack in which the password is assumed to be weak, *i.e.* part of a dictionary for which a brute force attack is feasible. A guessing attack works in two phases. In a first phase the attacker eavesdrops or interacts with one or several protocol sessions. In a second *offline* phase, the attacker tries each of the possible passwords on the data collected during the first phase. To resist against a guessing attack, the protocol must be designed such that the attacker cannot discover on the basis of the data collected whether his current guess of the password is the actual password or not. The definition below is due to M. 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 4.5 *The process* $\mathtt{new}\,\tilde{w}.A$ *is* resistant to guessing attacks *against* $\tilde{w}$ *if, for every process* B *such that* $\mathtt{new}\,\tilde{w}.A \stackrel{s}{\Rightarrow} \mathtt{new}\,\tilde{w}.B$ *(derivation obtained without renaming names in* $\tilde{w}$*), we have that* $\mathtt{new}\,\tilde{w}.(\phi(B) \mid \{^{\tilde{w}}/_{\tilde{x}}\}) \sim \mathtt{new}\,\tilde{w}'.\mathtt{new}\,\tilde{w}.(\phi(B) \mid \{^{\tilde{w}'}/_{\tilde{x}}\})$ *where* $\tilde{w}'$ *is a sequence of fresh names and* $\tilde{x}$ *a sequence of variables such that* $\tilde{x} \cap dom(\phi(B)) = \emptyset$.

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.

4.3.2 Privacy in electronic voting protocols

We report below on some of our recent efforts in using the equivalences of the applied pi calculus to model privacy-type properties of electronic elections [36, 6, 1]. Those properties can be expressed informally as follows:

- *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.

Vote-privacy. The privacy property aims to guarantee that the link between a given voter V and his vote v remains hidden. A classical device 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 suppose that at least two voters are honest. We denote the voters V_A and V_B and their votes a and b. We say that a voting protocol respects privacy whenever a process where V_A votes a and V_B votes b is equivalent to a process where V_A votes b and V_B votes a. Formally, we have defined privacy as follows:

Definition 4.6 (vote-privacy) *[36, 6] A voting protocol respects* vote-privacy *if*

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \;\approx\; S[V_A\{^b/_v\} \mid V_B\{^a/_v\}]$$

for all possible votes a *and* b.

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 *e.g.* if the vote is unanimous, or if all other voters reveal how they voted. 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 the context S.

Receipt-freeness. Similarly to privacy, receipt-freeness may be formalised as an 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 V_A^{ch} which has been modified: inputs and freshly generated names of base type (*i.e.* not channel type) are forwarded to the coercer on the 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 are forwarded but not the secret channel name on which the keys are received). The process $A^{\setminus out(ch,\cdot)}$ is as the process A, but hiding the outputs on the channel ch.

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 equivalence between two processes. Suppose the coercer's desired vote is c. Then we define receipt-freeness as follows:

Definition 4.7 (Receipt-freeness) *[36, 6] A voting protocol is* receipt-free *if there exists a closed plain process V' such that*

- $V'^{\setminus out(chc,\cdot)} \approx V_A\{^a/_v\}$,
- $S[V_A\{^c/_v\}^{chc} \mid V_B\{^a/_v\}] \approx S[V' \mid V_B\{^c/_v\}]$,

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 counterbalancing 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 we have performed. Therefore, we prefer to stick to Definition 4.7.

Coercion-resistance. *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 the voter 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. The definition is more involved and can be found in [6].

We have proved the intuitive relationships between the three properties:

Proposition 4.1 *[36, 6] Let V be a voting protocol.*

1. *If V is coercion-resistant (for a given set of honest authorities), then it also respects receipt-freeness (for the same set);*
2. *If V is receipt-free (for a given set of honest authorities), then it also respects privacy (for the same set).*

To validate our definitions, we have investigated three protocols from the literature:

1. The protocol by Fujioka, Okamoto, and Ohta [FOO92b] based on blind signatures and commitments;
2. The protocol by T. Okamoto [Oka96] based on trapdoor bit commitments. It was designed to be incoercible. However, Okamoto himself shows a flaw [Oka97]. According to him, one of the reasons why the voting scheme he proposed had such a flaw is that no formal definitions of receipt-freeness and coercion-resistance have been given when the concept of receipt-freeness has been introduced by J. Benaloh and D. Tuinstra [BT94].
3. The protocol by B. Lee *et al.* [LBD+04] based on designated verifier proofs of re-encryption.

Our results, fully described in [6], are summarised below:

Property	*Fujioka et al.*	*Okamoto et al.*	*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

Our reasonings about static equivalence and bisimulation in applied pi are rather informal. Having in mind these examples, we have developed better techniques for automating this reasoning (see Chapter 5 and Chapter 6 in which we will come back to these case studies).

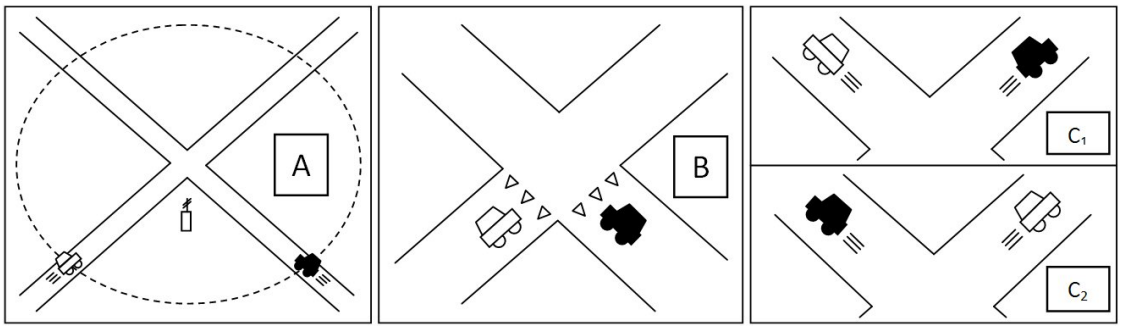
In the context of applied pi calculus, those definitions were the first formal definitions to model privacy-type properties in electronic voting. Since then, some other definitions have been proposed [BHM08]. Our definitions of privacy and receipt-freeness have also been reused and adapted to model privacy and receipt-freeness in on-line auction systems [DJP10]. We have also proposed an epistemic logic that is expressive enough to specify privacy [23]. Independently, a definition of coercion-resistance, based on an epistemic approach has also been proposed in [KT09]. This approach to model complex security properties seems to be promising. Epistemic logics are indeed well-suited to express general security properties in an intuitive and simple way.

4.3.3 Privacy in VANETs

To improve road safety, a vehicle-to-vehicle communication platform is currently being developed by consortia of car manufacturers and legislators [SAF10, Sta06]. To facilitate safety-critical applications there is a consensus that all vehicles must periodically broadcast a beacon message consisting of the vehicle's location, velocity, and identifier. Broadcasting this data several times per second raises privacy issues.

Fortunately, many of the envisioned applications do not need a real-world identifier, but can instead use a random identifier known as a pseudonym. However, long term tracking may still reveal the real-world identity of the driver. One can change pseudonym from time to time, but for this to have any effect the vehicles must change pseudonyms under the right circumstances. It seems preferable to change pseudonyms *e.g.* at intersections where several vehicles are close together and their paths unpredictable. This mimics the ubiquitous computing idea of a *mix-zone*, where beacon signals are turned off in a mixing area [BS03]. However, vehicles cannot turn off beacon messages since many accidents happen at intersections, hence the idea is to have all vehicles encrypt their beacon signals when inside the zone [FRF+07].

The formal privacy property aims to capture the fact that an attacker cannot track a vehicle. We assume that the attacker can listen on the entire network and hence on all public channels. Thus, in order to achieve privacy, we need to suppose the presence of at least two vehicles.



During their journey through the mix-zone, the vehicles will come in close enough proximity that the attacker is assumed unable to distinguish their location (part B of the figure). Before leaving the mix-zone, the vehicles change their pseudonyms leaving the attacker unable to determine if they leave according to part C_1 or part C_2 of the figure. Intuitively, we achieve privacy if an attacker cannot tell the two cases apart. Again, this can be expressed by an equivalence between two processes modelling the two different situations (see [15]). We will come back to this application in Chapter 6 to show how to analyse the resulting equivalence properties.

4.4 An existing tool: ProVerif

ProVerif constitutes a well-established automated protocol verifier based on Horn clauses resolution that allows for the verification of observational equivalence and of different trace-based security properties such as authenticity. This verifier has been mainly developed by B. Blanchet [Bla01]. Actually, this is the only tool that is able to establish observational

equivalence between processes written in applied pi calculus. After a brief description of the ProVerif tool, we will discuss the problem we encountered when we tried to use it to prove equivalences coming from our case studies.

4.4.1 A brief description

ProVerif takes as input processes written in a syntax close to the one described in Section 4.1. It is based on a representation of the protocol by Horn clauses. It can handle many different cryptographic primitives, including shared and public key cryptography (encryption and signatures), hash functions, and Diffie-Hellman key agreements, specified both as rewrite rules or as equations. It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. This result has been obtained thanks to some well-chosen approximations. This means that the verifier can give false attacks, but if it claims that the protocol satisfies some property, then the property is actually satisfied. It has been shown that the considered resolution algorithm terminates on a large class of protocols and when the tool cannot prove a property, it tries to reconstruct an attack. ProVerif can prove the following properties:

- secrecy (the adversary cannot obtain the secret) [Bla01];
- authentication [Bla02] and more generally correspondence properties [AB03];
- strong secrecy: the adversary does not see the difference when the value of the secret changes [Bla04];
- equivalences between processes that differ only by terms (so-called bi-processes) [BAF08].

4.4.2 Some limitations

The ProVerif tool has several limitations that prevents us to use it to analyse certain cryptographic protocols.

Equational theories. Even if ProVerif is able to deal with a variety of equational theories, this is still not sufficient to analyse some protocols. For instance, ProVerif fails to analyse protocols that rely on trapdoor bit commitments (*e.g.* electronic voting protocol by T. Okamoto [Oka97]).

Non-monotonic global states. Some of the over-approximations made by the tool do not work well with non-monotonic global 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. ProVerif does not model a state transition system, but rather a set of derivable facts representing attacker knowledge, together with the assumption that the attacker never forgets any fact. Actually, this restriction is problematic to model API (see Chapter 3) and also RFID protocols since the RFID tag often stores a secret from one session to another (see *e.g.* [BCdH10]).

Observational equivalence. The syntax of ProVerif contains the operator choice[_, _] which allows us to model a pair of processes that have the same structure and differ only in the choice of terms. Such a process is called a *bi-process*. Given a bi-process P, the process $\mathsf{fst}(P)$ is obtained by replacing all occurrences of choice$[M, M']$ with M. Similarly, $\mathsf{snd}(P)$ is obtained by replacing choice$[M, M']$ with M'. When a bi-process P satisfies what is called diff-equivalence, this implies that $\mathsf{fst}(P) \approx \mathsf{snd}(P)$.

However, diff-equivalence is a strong notion, and thus there are trivial equivalences which ProVerif is unable to prove. For instance, the equivalence

$$\mathtt{out}(c, a) \mid \mathtt{out}(c, b) \approx \mathtt{out}(c, b) \mid \mathtt{out}(c, a)$$

holds trivially since the processes are in fact structurally equivalent. However, the corresponding bi-process $\mathtt{out}(c, \mathrm{choice}[a, b]) \mid \mathtt{out}(c, \mathrm{choice}[b, a])$ does not satisfy diff-equivalence and therefore this equivalence cannot be proved by ProVerif. We already encountered this problem several times when modelling different kinds of cryptographic protocols from the literature, *e.g.* the e-voting protocol by Fujioka, Okamoto, and Ohta [FOO92b], or the Direct Anonymous Attestation protocol [BCC04] (more details in Chapter 6).

4.5 Conclusion and perspectives

In this chapter, we gave a brief description of the applied pi calculus [AF01] which has been used by many researchers to model and analyse security protocols in a variety of areas, *e.g.* certified email [AB03], election verifiability properties in e-voting protocols [KRS10], ... In such a calculus, the properties of the cryptographic primitives are modelled by means of an equational theory. This leads to a flexible calculus suitable to formalise a lot of security protocols. It is however not sufficient for instance to capture neighbourhood checks needed to model routing protocols (see Chapter 2). Hence, this calculus could be extended in several ways in order to model new applications. Actually, this calculus is particularly suitable to model privacy-type properties that are encountered in many applications. We present some of them in this chapter. Observational equivalence has also been used to model privacy in some other applications as well, *e.g.* unlinkability in RFID protocols [ACRR10, BCdH10], privacy-type properties in e-auction protocols [DJP10], ...

From the verification point of view, it is clearly important to develop procedures for deciding the security properties mentioned in this chapter. The following two chapters are entirely devoted to this goal. In Chapter 5, we describe several algorithms for deciding static equivalence. This allows us to cover a large class of equational theories. Then, in Chapter 6, we sum up our recent attempts to decide the more involved notion of observational equivalence.

From the modelling point of view, it would be interesting to better understand the relationship between observational equivalence and trace equivalence. We have shown that these two notions actually concide for determinate processes. We have also exhibited a syntactic class of determinate processes, namely the class of *simple processes* (see Chapter 6). Even if it seems quite natural to restrict our attention to determinate processes since most of the security protocols are indeed determinate, for modelling purposes, it is sometimes useful to consider processes outside this class. For instance, we may want to use private channels to distribute keys relying then on a composition result to establish the correctness of the entire protocol where keys are distributed via unreliable channels (*e.g.* [CC10]). In presence of

non-determinate processes, we first need to decide which notions of equivalence we want to use.

Originally, the notion of observational equivalence has been introduced as a proof technique for trace equivalence [AG97] leading to a co-inductive method for proving trace equivalence. Although bisimulation-based equivalences may be simpler to check than trace equivalences [KS83], in the context of cryptographic protocols, it seems sometimes easier to simply check trace equivalence, that is, equality of the set of execution traces (modulo some equivalence relation between traces). As future work, it would be interesting to study the complexity of both problems from both a theoretical and practical points of view.

Chapter 5

Static equivalence

Contents

Traditionally, the knowledge of the attacker is expressed in terms of *deducibility*. A message s (intuitively the secret) is said to be deducible from a set of messages ϕ, if an attacker is able to compute s from ϕ. To perform this computation, the attacker is allowed, for example, to decrypt deducible messages by deducible keys. However, deducibility is not always sufficient. Consider for example the case where a protocol participant sends over the network the encryption of one of the constants "yes" or "no" (*e.g.* the value of a vote). Deducibility is not the right notion of knowledge in this case, since both possible values ("yes" and "no") are indeed "known" to the attacker. In this case, a more adequate form of knowledge is *indistinguishability* (*e.g.* [AC06]): is the attacker able to distinguish between two transcripts of the protocol, one running with the value "yes" and the other one running with the value "no"?

The two notions of knowledge that we consider do not take into account the dynamic behaviour of the protocol. Nevertheless, in order to establish that two dynamic behaviours of

a protocol are indistinguishable, an important subproblem is to establish indistinguishability between the sequences of messages generated by the protocol. As we have seen in Chapter 4, indistinguishability between sequences of messages, also called static equivalence in the applied pi calculus framework, plays an important role in the study of guessing attacks as well as for privacy-type properties.

In this chapter, we consider both deducibility and indistinguishability. The two notions are formally recalled in Section 5.1. Then, we provide several algorithms for deciding these two notions for a wide variety of equational theories. First, we consider convergent equational theories and we present two algorithms based on a saturation procedure. These algorithms have been implemented in the tools YAPA and KiSs (see Section 5.2). In Section 5.3, we present a general setting for solving deducibility and indistinguishability for an important class (called *monoidal*) of equational theories involving AC operators. Lastly, we show that existing decidability results can be easily combined for any disjoint equational theories: if the deducibility and indistinguishability relations are decidable for two disjoint theories, they are also decidable for their union. This result is stated in Section 5.4. As a consequence of all these results, new decidability and complexity results can be obtained for many relevant equational theories. In Section 5.5, we give a list of relevant equational theories for which deduction and static equivalence have been studied by us or others. This gives a (hopefully) complete picture of existing results in this area.

5.1 Definitions

In this section, we formally state the two problems (deduction problem and static equivalence problem) that are studied along this chapter.

5.1.1 Deduction

Given a frame ϕ that represents the information available to an attacker, we may ask whether a given ground term M may be deduced from ϕ. Intuitively, the deducible messages are the messages of ϕ and the names that are not protected in ϕ, closed by equality in E and closed by application of function symbols.

Definition 5.1 (recipe) *Let M be a ground term and $\phi = \mathtt{new}\,\tilde{n}.\sigma$ be a frame. A* recipe of M in ϕ modulo E *is a term $\zeta \in \mathcal{T}(\mathcal{F},\mathcal{N} \cup \mathcal{X})$ such that $fn(\zeta) \cap \tilde{n} = \emptyset$ and $\zeta\sigma =_{\mathsf{E}} M$. In such a case, we say that M is* deducible *from ϕ modulo E, and we write $\phi \vdash_{\mathsf{E}} M$.*

When $\mathtt{new}\ \tilde{n}.\sigma \vdash_{\mathsf{E}} M$, this means that any occurrence of names from $\tilde{n}$ in M is bound by $\mathtt{new}\ \tilde{n}$. So $\mathtt{new}\ \tilde{n}.\sigma \vdash_{\mathsf{E}} M$ could be formally written $\mathtt{new}\ \tilde{n}.(\sigma \vdash_{\mathsf{E}} M)$. We reuse the notation $\vdash$ first introduced in Chapter 2. Here, the abilities of the attacker to deduce messages are expressed by the means of an equational theory instead of an inference system. Actually, the relation described above can be axiomatized by the following inference rules:

$$\frac{}{\mathtt{new}\ \tilde{n}.\sigma \vdash_{\mathsf{E}} M}\ \text{if } \exists x \in dom(\sigma) \text{ such that } x\sigma = M \qquad \frac{}{\mathtt{new}\ \tilde{n}.\sigma \vdash_{\mathsf{E}} s}\ s \in \mathcal{N} \setminus \tilde{n}$$

$$\frac{\phi \vdash_{\mathsf{E}} M_1 \quad \ldots \quad \phi \vdash_{\mathsf{E}} M_\ell}{\phi \vdash_{\mathsf{E}} \mathsf{f}(M_1,\ldots,M_\ell)}\ \mathsf{f} \in \Sigma \qquad \frac{\phi \vdash_{\mathsf{E}} M}{\phi \vdash_{\mathsf{E}} M'}\ M =_{\mathsf{E}} M'$$

Deduction problem modulo the equational theory E built over $\mathcal{F}$.

Entries: A frame ϕ and a ground term M (both built over $\mathcal{F}$)
Question: $\phi \vdash_{\mathsf{E}} M$?

5.1.2 Static equivalence

Deduction does not always suffice for expressing the knowledge of an attacker, as discussed in the introduction of this chapter. Sometimes, the attacker can deduce exactly the same set of terms from two different frames but he could still be able to tell the difference between these two frames. This indistinguishability relation has been formally defined in Chapter 4 (see Definition 4.1). Here, we give another characterization. Let E be an equational theory built over $\mathcal{F}$. We define $\mathsf{Eq}_{\mathsf{E}}(\phi)$ to be the set of equations satisfied by the frame ϕ.

$$\mathsf{Eq}_{\mathsf{E}}(\phi) = \{(M, N) \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X}) \mid (M =_{\mathsf{E}} N)\phi\}.$$

We write $\psi \models \mathsf{Eq}_{\mathsf{E}}(\phi)$ if $(M =_{\mathsf{E}} N)\psi$ for any $(M, N) \in \mathsf{Eq}_{\mathsf{E}}(\phi)$.

Checking for static equivalence is clearly equivalent to checking whether each of the two frames under consideration satisfies the equalities of the other frame. Hence, we have that:

$$\phi_1 \sim_{\mathsf{E}} \phi_2 \quad \Leftrightarrow \quad \phi_2 \models \mathsf{Eq}_{\mathsf{E}}(\phi_1) \text{ and } \phi_1 \models \mathsf{Eq}_{\mathsf{E}}(\phi_2).$$

Static equivalence problem modulo the equational theory E built over $\mathcal{F}$.

Entries: Two frames ϕ_1 and ϕ_2 (both built over $\mathcal{F}$)
Question: $\phi_1 \sim_{\mathsf{E}} \phi_2$?

5.2 Convergent equational theories

First, we have proposed a generic procedure for deducibility and static equivalence that takes as input any convergent theory, *i.e.* any equational theory described by a finite convergent rewrite system [22]. Since the problem of checking deducibility and static equivalence is undecidable for this class [AC06], unfailing termination can not be guaranteed in general. To address this issue and turn our algorithm into a decision procedure for a given equational theory, we provide several criteria. This allows us to obtain a generic algorithm for all the convergent theories shown to be decidable in [AC06] (with different algorithms).

5.2.1 Applications

We give in this section several theories that are in the scope of the results stated and proved in [22, 20, 3]. We are particularly interested by those allowing us to model electronic voting protocols, *e.g.* blind signatures, trapdoor commitments, ...

Subterm convergent equational theories. A subterm convergent theory is an equational theory induced by a finite set of equations of the form $u = v$ where v is either a subterm of u or a constant, and such that the associated rewrite system is convergent.

Blind signature. In [6], we used this theory in order to model blind signatures for the analysis an e-voting protocol [FOO92b].

$$\mathsf{unblind}(\mathsf{blind}(x,y),y) = x \qquad \mathsf{checksign}(\mathsf{sign}(x,y),\mathsf{pk}(y)) = x$$
$$\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) = \mathsf{sign}(x,z)$$

Homomorphism encryption. This theory represents an encryption scheme with a homomorphism property.

$$\begin{array}{rclrcl} \mathsf{senc}(\langle x,y\rangle,z) & = & \langle \mathsf{senc}(x,z),\mathsf{senc}(y,z)\rangle & \mathsf{proj}_1(\langle x,y\rangle) & = & x \\ \mathsf{sdec}(\langle x,y\rangle,z) & = & \langle \mathsf{sdec}(x,z),\mathsf{sdec}(y,z)\rangle & \mathsf{proj}_2(\langle x,y\rangle) & = & y \\ & & & \mathsf{sdec}(\mathsf{senc}(x,y),y) & = & x \end{array}$$

Trapdoor commitment. The following equational theory E_{td} is a model for trapdoor commitment.

$$\begin{array}{ll} \mathsf{open}(\mathsf{td}(x,y,z),y) = x & \mathsf{td}(x_2,\mathsf{f}(x_1,y,z,x_2),z) = \mathsf{td}(x_1,y,z) \\ \mathsf{open}(\mathsf{td}(x_1,y,z),\mathsf{f}(x_1,y,z,x_2)) = x_2 & \mathsf{f}(x_2,\mathsf{f}(x_1,y,z,x_2),z,x_3) = \mathsf{f}(x_1,y,z,x_3) \end{array}$$

We encountered this equational theory when studying the electronic voting protocols described in [Oka96]). The term $\mathsf{td}(m,r,td)$ models the commitment of the message m under the key r using an additional trapdoor td. Such a commitment scheme allows a voter who has performed a commitment to open it in different ways using its trapdoor.

Designated verifier proofs (DVP) of re-encryption. We came across this theory when studying the electronic voting protocol described in [LBD+04].

$$\mathsf{adec}(\mathsf{penc}(x,\mathsf{pk}(y),z),y) = x$$
$$\mathsf{renc}(\mathsf{penc}(x,\mathsf{pk}(y),z_1),z_2) = \mathsf{penc}(x,\mathsf{pk}(y),\mathsf{f}(z_1,z_2))$$
$$\mathsf{checkdvp}(\mathsf{dvp}(x,\mathsf{renc}(x,z),z,\mathsf{pk}(y)),x,\mathsf{renc}(x,z),\mathsf{pk}(y)) = \mathsf{ok}$$
$$\mathsf{checkdvp}(\mathsf{dvp}(x_1,x_2,z,y),x_1,x_2,\mathsf{pk}(y)) = \mathsf{ok}$$

A re-encryption of a ciphertext (obtained using a randomised encryption scheme) changes the random coins, without changing or revealing the plaintext. A DVP of the re-encryption proves that the two ciphertexts contain indeed the same plaintext. However, a designated verifier proof only convinces one intended person, *e.g.* the voter, that the re-encrypted ciphertext contains the original plaintext. In particular this proof cannot be used to convince the coercer.

5.2.2 A generic procedure

Let E be a convergent equational theory and $\mathcal{R}$ its associated convergent rewrite system. We denote by $t{\downarrow_{\mathcal{R}}}$ (or simply $t{\downarrow}$ when $\mathcal{R}$ is clear from the context) the normal form of t. The core of the procedure that is fully described in [22] consists of a set of transformation rules used to saturate a state (Φ,Ψ) made up of a finite set of facts, denoted by $M \rhd t$, and a finite set of quantified equations, denoted by $\forall \tilde{z}.M \bowtie N$.

Given a frame φ, the result of the saturation is a state (Φ,Ψ) such that:

- Ψ characterises all the equations that are true in φ. More precisely, we have that:

$$(M =_{\mathsf{E}} N)\varphi \;\Leftrightarrow\; \Psi \models M \bowtie N$$

  where $\Psi \models M \bowtie N$ means that the equation $M \bowtie N$ is a consequence of Ψ in the usual first-order logic with equality axioms for the relation $\bowtie$ (*i.e.* reflexivity, symmetry, transitivity and compatibility with symbols in $\mathcal{F}$ that are public).

- Φ characterises all the deducible terms of φ. More precisely, we have that:

$$\varphi \vdash_{\mathsf{E}} t \;\Leftrightarrow\; \Phi \vdash_{\emptyset} t{\downarrow_{\mathcal{R}}}$$

Decision procedures for deduction and static equivalence modulo E follow from the result described above.

Algorithm for deduction. Let φ be an initial frame and t be a ground term. The procedure for checking $\varphi \vdash_{\mathsf{E}} t$ runs as follows:

1. Apply the transformation rules on φ to obtain (if any) a saturated state (Φ, Ψ);
2. Return *yes* if $\Phi \vdash t{\downarrow}$ (that is, the $\mathcal{R}$-reduced form of t is syntactically deducible from Φ); otherwise return *no*.

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

1. Apply the transformation rules on φ_1 and φ_2 to obtain (if possible) two saturated states (Φ_1, Ψ_1) and (Φ_2, Ψ_2);
2. For $\{i, j\} = \{1, 2\}$ and for every equation $(\forall z_1, \ldots, z_\ell . M \bowtie N)$ in Ψ_i, check that $M\varphi_j =_{\mathsf{E}} N\varphi_j$, that is, in other words, $(M\varphi_j){\downarrow} = (N\varphi_j){\downarrow}$;
3. If so return *yes*; otherwise return *no*.

5.2.3 Non-failure and termination

Note that (unfailing) termination cannot be guaranteed in general since the problem of checking deducibility and static equivalence is undecidable, even for convergent theories [AC06]. To address this issue and turn our algorithm into a decision procedure for a given convergent theory, we provide two criteria (see [22] for more details).

A syntactic criterion to ensure non-failure. We have proved that our algorithm never fails for *layered convergent* theories. This criterion is enjoyed in particular by any subterm convergent theory, as well as the theories of blind signature and homomorphic encryption. Termination often follows from a simple analysis of the rules of the algorithm.

A semantic criterion to ensure termination. We also provide a semantic criterion that more generally explains why our procedure succeeds on theories previously known to be decidable [AC06]. This criterion intuitively states that the set of deducible terms from any initial frame φ should be equivalent to a set of *syntactically* deducible terms. Provided that failures are prevented and assuming a *fair* strategy for rule application, we prove that this criterion is a necessary and sufficient condition for our procedure to terminate.

Together with the syntactic criterion described to prevent non-failure, this criterion allows us to prove decidability of deduction and static equivalence for layered convergent theories that also belong to the class of locally stable theories defined in [AC06]. As a consequence, our procedure always saturates for the theories of blind signature and homomorphic encryption since those theories are layered and have been proved locally stable [AC06]. Other examples of layered convergent theories enjoying this criterion can be found in [AC06] (*e.g.* a theory of addition). While in [AC06] the decision algorithm needs to be adapted for each theory, we propose a single (and efficient) algorithm that ensures a unified treatment of all these theories.

5.2.4 More equational theories

As shown by the following example (from [20]), the procedure described in the previous section may fail.

Example 5.1 *Consider the convergent theory* $\mathsf{E}_{\mathsf{mal}}$ *described below:*

$$\mathsf{E}_{\mathsf{mal}} = \{\mathsf{sdec}(\mathsf{senc}(x,y),y) = x,\ \mathsf{mal}(\mathsf{senc}(x,y),z) = \mathsf{senc}(z,y)\}.$$

The mal *function symbol allows one to arbitrarily change the plaintext of an encryption. Such a malleable encryption is not realistic. It is only used for illustrative purposes. Let* $\varphi = \{\mathsf{senc}(s,k)\}$ *where* s *and* k *are private constants.*

Among the terms that are deducible from φ*, there are* $\mathsf{senc}(a,k)$, $\mathsf{senc}(b,k)$, *... and more generally all the terms of the form* $\mathsf{senc}(m,k)$ *where* m *is any term that is deducible from* φ*. Since our procedure represents the deducible terms by a finite set of ground terms (this finite set of terms can be seen as a basis to generate all the deducible terms), it is easy to see that the procedure described above can not reach a saturated state when we consider the theory* $\mathsf{E}_{\mathsf{mal}}$*.*

The same problem happens for the trapdoor commitment theory. Hence, in [20, 3], we have proposed a symbolic representation of deducible terms which manipulates terms that are not necessarily ground and facts with side conditions. For instance, consider again the frame $\varphi = \{^{\mathsf{senc}(s,k)}/_{x_1}\}$ and the theory $\mathsf{E}_{\mathsf{mal}}$, we will have the following fact to deal with the situation described in Example 5.1

$$\mathsf{mal}(x_1, X) \rhd \mathsf{senc}(x,k) \mid X \rhd x$$

Intuitively, this fact says that $\mathsf{senc}(x\sigma,k)$ is deducible as soon as $x\sigma$ is deducible. Moreover, assuming that $x\sigma$ is deducible by using the recipe ζ, the recipe associated to $\mathsf{senc}(x\sigma,k)$ is $\mathsf{mal}(x_1,\zeta)$. The procedure we have proposed in [20, 3] does not fail. However, termination is not ensured in general. We provide a generic method for proving termination that we instantiate on several examples. This allows us to derive PTIME decision procedures for deduction and static equivalence for the subterm convergent equational theories, malleable encryption, trapdoor commitment, and blind signature. We also guarantee termination for homomorphic encryption by relying on a *fair* saturation strategy.

5.2.5 Implementations: YAPA and KiSs

We give a brief description of the tools that correspond to the implementations of the procedures described in the previous section.

YAPA. YAPA (Yet Another Protocol Analyzer) is an Ocaml implementation of the saturation procedure presented in Section 5.2.2 with several optional optimizations. It can be freely downloaded [Bau08] together with a brief manual and examples. The tool takes as input an equational theory described by a finite convergent rewrite system, as well as frame definitions and queries.

We have conducted several experiments for various equational theories and found that YAPA provides an efficient way to check static equivalence and deducibility (few seconds). Those examples are available at:

`http://www.lsv.ens-cachan.fr/~baudet/yapa/index.html`.

KiSs. A C++ implementation of the procedures fully described in [20, 3] is provided in the KiSs (Knowledge in Security protocols) tool [Cio09]. Actually, the tool implements the procedure with some optimizations. This makes the procedure terminate in polynomial time for subterm convergent equational theories, blind signature, malleable encryption, trapdoor commitment, and homomorphic encryption.

The performances of the KiSs tool are comparable to the YAPA tool and on most examples the tool terminates in less than a second. In [22], we presented a family of contrived example 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. More details about the KiSs tool are available on line:

`http://www.lsv.ens-cachan.fr/~ciobaca/kiss/`.

Related work. In comparison with the ProVerif tool [Bla01, BAF08], here instrumented to check static equivalences, our test samples suggest a running time faster for YAPA and KiSs. Also we did not succeed in making ProVerif terminate on the theories of homomorphic encryption and trapdoor commitment. Of course, these results are not entirely surprising given that ProVerif is tailored for the more general (and difficult) problem of protocol (in)security under active attackers. In particular ProVerif's initial preprocessing of the rewrite system appears more substantial than ours and does not terminate on the theory $\mathsf{E}_{\mathsf{hom}}$. However, ProVerif handles some non-convergent theories such as commutativity and the equation $\mathsf{exp}(\mathsf{exp}(\mathsf{g}, x), y) = \mathsf{exp}(\mathsf{exp}(\mathsf{g}, y), x)$ which can be used as a basic model of Diffie-Hellman. These theories are out of scope of our tools YAPA and KiSs.

Recently, new polynomial time algorithms for deduction and static equivalence under subterm convergent equational theories have been proposed [CBC10]. Their procedures have many concepts in common with the algorithms presented in this section and have not yet been implemented. However, they achieve a significantly better asymptotic complexity.

5.3 Monoidal equational theories

This class of theories has been introduced by F. Baader [Baa89] and W. Nutt [Nut90]. This class captures many theories with associative and commutative properties (AC), which are known to be difficult to deal with. In [30, 2], we have proposed a general schema for deciding deduction and static equivalence. This schema has to be filled with procedures for linear equations in order to yield complete algorithms. Such algorithms strongly depend on the structure of the semiring associated to a monoidal theory. We will see that algebra provides useful techniques and results to fill in this gap.

5.3.1 Definitions

In this section, we first define monoidal theories and then give examples.

Definition 5.2 (monoidal theory) *A theory* E *over* $\mathcal{F}$ *is called* monoidal *if it satisfies the following properties:*

1. *The signature* $\mathcal{F}$ *contains a binary function symbol* $+$ *and a constant symbol* 0*, and all other function symbols in* $\mathcal{F}$ *are unary.*
2. *The symbol* $+$ *is associative-commutative with unit* 0*,* i.e. *the equations* $x + (y + z) = (x + y) + z$*,* $x + y = y + x$ *and* $x + 0 = x$ *are in* E*.*
3. *Every unary function symbol* $\mathsf{h} \in \mathcal{F}$ *is an endomorphism for* $+$ *and* 0*, i.e.* $\mathsf{h}(x + y) = \mathsf{h}(x) + \mathsf{h}(y)$ *and* $\mathsf{h}(0) = 0$ *are in* E*.*

Suppose $+$ is a binary function symbol and 0 is nullary. Moreover assume that the other symbols, *i.e* $-$, h, are unary symbols. The equational theories below are monoidal.

- The theory ACU over $\mathcal{F} = \{+, 0\}$ which consists of the axioms of associativity and commutativity with unit 0.
- The theory ACUN (*exclusive or*) over $\mathcal{F} = \{+, 0\}$ which consist of the axioms (AC) and (U) with in addition Nilpotency (N) $\quad x + x = 0$.
- The theory AG (*Abelian groups*) over $\mathcal{F} = \{+, -, 0\}$ which is generated by the axioms (AC), (U) and $x + -(x) = 0$ (Inv). Indeed, the equations $-(x + y) = -(x) + -(y)$ and $-0 = 0$ are consequences of the others.
- The theories ACUh, ACUNh over $\mathcal{F} = \{+, \mathsf{h}, 0\}$ and AGh over $\mathcal{F} = \{+, -, \mathsf{h}, 0\}$: these theories correspond to the ones described above extended by the homomorphism laws (h) for the symbol h, *i.e.* $\mathsf{h}(x + y) = \mathsf{h}(x) + \mathsf{h}(y)$ and $\mathsf{h}(0) = 0$ (if it is not a consequence of the other equations).

Monoidal theories have an algebraic structure close to rings except that elements might not have an additive inverse. Such a structure is called a *semiring*.

Definition 5.3 (semiring) *A* semiring *is a set* $\mathcal{S}$ *(called the* universe *of the semiring) with distinct elements* 0 *and* 1 *that is equipped with two binary operations* $+$ *and* $\cdot$ *such that* $(\mathcal{S}, +, 0)$ *is a commutative monoid,* $(\mathcal{S}, \cdot, 1)$ *is a monoid, and the following identities hold for all* $\alpha, \beta, \gamma \in \mathcal{S}$*:*

- $(\alpha + \beta) \cdot \gamma = \alpha \cdot \gamma + \beta \cdot \gamma$ *(right distributivity)*
- $\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$ *(left distributivity)*
- $0 \cdot \alpha = \alpha \cdot 0 = 0$ *(zero laws).*

It has been shown in [Nut90] that for any monoidal theory E there exists a corresponding semiring $\mathcal{S}_{\mathsf{E}}$. For instance, we have that

1. The semiring $\mathcal{S}_{\mathsf{ACU}}$ is isomorphic to $\mathbb{N}$, the semiring of natural numbers.
2. The semiring $\mathcal{S}_{\mathsf{ACUN}}$ consists of the two elements 0 and $\mathbf{1}$ and we have $0+\mathbf{1} = \mathbf{1}+0 = \mathbf{1}$, $0+0 = \mathbf{1}+\mathbf{1} = 0$, $0 \cdot 0 = \mathbf{1} \cdot 0 = 0 \cdot \mathbf{1} = 0$, and $\mathbf{1} \cdot \mathbf{1} = \mathbf{1}$. Hence, $\mathcal{S}_{\mathsf{ACUN}}$ is isomorphic to the commutative ring (field) $\mathbb{Z}/2\mathbb{Z}$.
3. The semiring $\mathcal{S}_{\mathsf{AGh}}$ is isomorphic to $\mathbb{Z}[\mathsf{h}]$ which is a commutative ring.

5.3.2 Reduction results

We have shown that solving a deduction problem can be reduced to solving a linear system of equations in the corresponding semiring.

Theorem 5.1 *[30, 2] Let E be a monoidal theory and $\mathcal{S}_{\mathsf{E}}$ be its associated semiring. Deduction in E is reducible in polynomial time to the following problem:*

Entries*: A matrix A over $\mathcal{S}_{\mathsf{E}}$ of size $\ell \times m$ and a vector b over $\mathcal{S}_{\mathsf{E}}$ of size ℓ*
Question*: Does there exist X (a vector over $\mathcal{S}_{\mathsf{E}}$ of size ℓ) such that $X \cdot A = b$?*

Note that when $\mathcal{S}_{\mathsf{E}}$ is commutative, this problem is equivalent to the problem of deciding whether there exists Y such that $A^{\mathsf{T}} \cdot Y = b^{\mathsf{T}}$, *i.e* whether b^{T} is in the image of A^{T} where M^{T} is the transpose of M.

Example 5.2 *Consider the theory* ACUN *(exclusive or) and the term $M = n_1 + n_3$. Let $\phi = \mathtt{new}\, n_1, n_2, n_3.\{^{n_1+n_2}/_{x_1}, {}^{n_2+n_3}/_{x_2}\}$. We have that:*

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

The equation $X \cdot A = b$ has a solution over $\mathbb{Z}/2\mathbb{Z}$: $(1, 1)$. The term M is deducible from ϕ by using the recipe $x_1 + x_2$.

As a consequence, decidability/complexity results for deduction can be deduced from decidability/complexity results for solving linear system of equations over semirings (see Section 5.3.3).

We have shown that deciding whether two frames are equivalent can be reduced to deciding whether two matrices satisfy the same set of equalities.

Theorem 5.2 *[30, 2] Let* E *be a monoidal theory and* $\mathcal{S}_\mathsf{E}$ *be its associated semiring. Static equivalence in* E *is reducible in polynomial time to the following problem:*
Entries*: Two matrices* A_1 *and* A_2 *over* $\mathcal{S}_\mathsf{E}$ *of size* $\ell \times m$
Question*: Does the following equality holds?*

$$\{(X,Y) \in \mathcal{S}_\mathsf{E}^\ell \times \mathcal{S}_\mathsf{E}^\ell \mid X \cdot A_1 = Y \cdot A_1\} = \{(X,Y) \in \mathcal{S}_\mathsf{E}^\ell \times \mathcal{S}_\mathsf{E}^\ell \mid X \cdot A_2 = Y \cdot A_2\}$$

Thanks to Theorem 5.2, we give a way to decide static equivalence in monoidal equational theories provided we can decide whether two sets of linear equations over $\mathcal{S}_\mathsf{E}$ have the same set of solutions. Actually, when $\mathcal{S}_\mathsf{E}$ is a ring or when we can extend the semiring $\mathcal{S}_\mathsf{E}$ into a ring $\mathcal{R}_\mathsf{E}$, the static equivalence problem is equivalent to the problem of deciding whether the equality

$$\{Z \in \mathcal{R}_\mathsf{E}^\ell \mid Z \cdot A_1 = 0\} = \{Z \in \mathcal{R}_\mathsf{E}^\ell \mid Z \cdot A_2 = 0\}$$

holds. When $\mathcal{R}_\mathsf{E}$ is commutative, it is equivalent to deciding whether $\mathsf{Ker}(A_1) = \mathsf{Ker}(A_2)$, where $\mathsf{Ker}(M)$ denotes the kernel of the matrices M, *i.e.* the set $\{X \mid M \cdot X = 0\}$.

5.3.3 Applications

In this section, we show that several interesting monoidal equational theories induce a ring or a semiring for which solving linear systems or checking for equalities of sets of solutions of linear systems are decidable. This allows us to retrieve many existing decidability and complexity results and also some new ones (see Table in Section 5.5).

Theory ACU**.** This equational theory is the simplest monoidal theory. The semiring corresponding to this theory is $\mathbb{N}$ whereas its associated ring is $\mathbb{Z}$. Since the problem of solving linear equations over $\mathbb{N}$ is NP-complete, we obtain that deduction is a NP-complete problem. Thanks to our algebraic characterization, the static equivalence problem can be solved in polynomial time [Sch86].

Theory ACUN **(Exclusive Or).** The semiring corresponding to this equational theory is the finite field $\mathbb{Z}/2\mathbb{Z}$. Deduction and static equivalence are both decidable in polynomial time for this theory.

Theory AG **(Abelian Groups).** The semiring associated to this equational theory is in fact a ring, namely the ring $\mathbb{Z}$ of all integers. There exist several algorithms to compute solutions of linear equations over $\mathbb{Z}$ and to compute a base of the set of solutions (see for instance [Sch86]). Hence, we easily deduce that both problems are decidable in PTIME.

Theories ACUh, ACUNh **and** AGh**.** The semiring associated to ACUh is $\mathbb{N}[\mathsf{h}]$, the semiring of polynomials in one indeterminate over $\mathbb{N}$, whereas the ring associated to ACUh is $\mathbb{Z}[\mathsf{h}]$. For the theory ACUNh (resp. AGh) the associated semiring is $\mathbb{Z}/2\mathbb{Z}[\mathsf{h}]$ (resp. $\mathbb{Z}[\mathsf{h}]$).

1. ACUh and AGh: Deciding static equivalence for both these theories is reducible to the problem of deciding whether $\mathsf{Ker}(A) = \mathsf{Ker}(B)$ where A and B are matrices built over $\mathbb{N}[\mathsf{h}]$ in the case of ACUh and $\mathbb{Z}[\mathsf{h}]$ in the case of AGh. This problem has been solved by F. Baader to obtain a unification algorithm for the theory AGh (see [Baa93]). This is done by the help of Gröbner Base methods in a more general setting. Actually, he provides an algorithm even in the case of several commuting homomorphisms.

2. ACUNh: Deciding static equivalence in ACUNh is reducible to the problem of deciding whether $\mathsf{Ker}(A) = \mathsf{Ker}(B)$ where A and B are matrices built over $\mathbb{Z}/2\mathbb{Z}[\mathsf{h}]$. This is achieved in [LLT06] by using an automata-theoretic approach.

5.4 Combination for disjoint theories

In [32], we provide a general combination result for both deduction and static equivalence: if the deducibility and indistinguishability relations are decidable for two disjoint theories E_1 and E_2 (that is, the equations of E_1 and E_2 do not share any signature symbol), they are also decidable for their union $\mathsf{E}_1 \cup \mathsf{E}_2$. Our combination results follow the approach of Y. Chevalier and M. Rusinowitch [CR05, CR08b], who show how to combine decision algorithms for the deducibility problem in presence of an active attacker. Our procedures also rely on combination algorithms for solving unification problems modulo E [SS89, BS96], and we partly reuse the techniques introduced by F. Baader and K. Schulz to combine constraint solvers [BS98].

Theorem 5.3 *[32, 2] Let E_1 and E_2 be two equational theories built over $\mathcal{F}_1$ (resp. $\mathcal{F}_2$) such that $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ and for which the word problem is decidable. If deduction is decidable for E_1 and E_2, then deduction is decidable for $\mathsf{E}_1 \cup \mathsf{E}_2$.*

Decidability of the word problem for the theories E_1 and E_2 allows us to ensure that we are able to decide whether two (ground) terms are equal or not modulo $\mathsf{E}_1 \cup \mathsf{E}_2$. This property is satisfied by all the theories that can be represented by a convergent (possibly modulo AC) rewrite system.

Theorem 5.4 *[32, 2] Let E_1 and E_2 be two equational theories built over $\mathcal{F}_1$ (resp. $\mathcal{F}_2$) such that $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$. If deduction and static equivalence are decidable for E_1 and E_2, then static equivalence is decidable for the equational theory $\mathsf{E}_1 \cup \mathsf{E}_2$.*

Actually, we show that whenever static equivalence is decidable for E_1 and E_2 and deduction is decidable for $\mathsf{E}_1 \cup \mathsf{E}_2$, then static equivalence is decidable for $\mathsf{E}_1 \cup \mathsf{E}_2$. Thanks to our combination result for deduction (Theorem 5.3), we know it is sufficient for deduction to be decidable for E_1 and E_2. Note that the decidability of $\vdash_{\mathsf{E}_i}$ is not necessarily a consequence of the decidability of $\sim_{\mathsf{E}_i}$. The encoding proposed in [AC06] works only when there exists a free function symbol in $\mathcal{F}_i$.

This combination result allows us to combine any existing decidability results for deduction and static equivalence provided the signatures of the equational theories are disjoint. Those combination algorithms can be applied for instance to combine a monoidal equational theory with any other equational theory for which deduction and static equivalence are known to be decidable.

5.5 Conclusion and perspectives

This chapter provides many decidability and complexity results for deduction and static equivalence, two formal representations for knowledge in the analysis of security protocols. First, we propose several generic decision procedures together with their implementations. These procedures work for a large class of convergent equational theories. Second, we propose a

general setting for an important class of equational theories with associative and commutative properties. Lastly, we show that existing decidability results can be combined for any disjoint equational theories. As a consequence, new decidability and complexity results can be obtained for many relevant equational theories. We sum up in the table below the existing results. We only mention those concerning equational theories that have been introduced in this chapter.

Theory E	Deduction	Static Equivalence
subterm convergent	decidable [22, 20], PTIME [AC06, CBC10] & [3]	
blind signature	decidable [AC06] &[22], PTIME [3]	
homomorphic encryption	decidable [AC06] & [22, 20, 3]	
malleable encryption	PTIME [3]	
trapdoor commitment	PTIME [BRC09] & [3]	
dvp of re-encryption	PTIME [BRC09]	
ACU	NP-complete	decidable [AC06] PTIME [30, 2]
ACUN	PTIME [CKRT03a] & [30, 2]	PTIME [AC06] & [30, 2]
AG	PTIME [CKRT03a] & [30, 2]	PTIME [30, 2]
ACUh	NP-complete [LLT05]	decidable [30, 2]
ACUNh	PTIME [12]	decidable [30, 2]
AGh	PTIME [12]	decidable [30, 2]
$\mathsf{AGh}_1 \ldots \mathsf{h}_n$	decidable [30, 2]	
subterm conv. $\uplus$ ACUN	PTIME [32, 2]	
subterm conv. $\uplus$ AG	PTIME [32, 2]	

The tool KiSs supports several equational theories for which the procedure implemented in YAPA fails. Conversely the procedure implemented in YAPA is guaranteed to terminate (without failure) for classes of theories that are not considered by the procedure implemented in KiSs. It would be interesting to compare the techniques and possibly to combine them in order to capture more theories. As further work, we would like to extend YAPA and/or KiSs to theories with associative and commutative operators. A first possibility would be to implement the decidability result presented in Section 5.3 for monoidal theories (that include many theories with associative and commutative operators) and to combine the two procedures using the combination theorems presented in Section 5.4. However, it seems much more efficient to integrate associativity and commutativity directly and this could even open the way to a more powerful combination technique.

Another interesting and probably challenging problem is to extend the combination results presented in Section 5.4 for non disjoint theories. We might use for example a notion of hierarchy between theories like in [CR08b]. A first result in this direction has been obtained by S. Kremer *et al.* [KMT11]. Their method allows one in some cases to simplify

the task of deciding static equivalence in a multi-sorted setting, by removing a symbol from the term signature and reducing the problem to several simpler equational theories. In particular, this technique allows them to decide static equivalence for bilinear pairings. This constitutes a first step towards finding generic criteria. However, some fragments of the modular exponentiation theory such as the Diffie-Hellman one, *i.e.* the axioms $\mathsf{exp}(x, 1) = x$ and $\mathsf{exp}(\mathsf{exp}(x, y), z) = \mathsf{exp}(x, y \times z)$ where $\times$ is an Abelian group operator, or the equation $\mathsf{exp}(x, y) \cdot \mathsf{exp}(x, z) = \mathsf{exp}(x, y + z)$, are out of the scope of their combination result.

Lastly, deduction and static equivalence are static notions. They do not take into account the dynamic behaviour of the underlying protocols. Even if these notions play an important role for the analysis of security protocols in presence of an active attacker, it remains challenging to obtain decidability results for the active case, especially in presence of algebraic properties. It would be also interesting to extend our combination result to the active case. Such kind of results already exists for trace-based security properties [CR08b], but no combination result exist for equivalence-based security properties.

Chapter 6

Observational equivalence

Contents

Observational equivalence is crucial when specifying privacy-type properties like anonymity that states that an observer cannot see the difference when A is talking and when B is talking (see Chapter 4). The goal of this chapter is to show how to verify this kind of security properties in presence of an active attacker. In Chapter 5, we only focus on the indistinguishability problem between two sequences of messages generated by the protocol. Thus, we do not take into account the dynamic behaviour of the underlying protocol. Here, we consider an active attacker who may interact with the protocol. In Section 4.4, we have seen that the ProVerif tool makes some progress in this direction. However the tool has several limitations that prevent us from establishing the equivalences needed to establish privacy on most of our examples.

First, we propose a technique for expanding the scope of the ProVerif tool (see Section 6.1). Then, following the constraint solving approach developed in Chapter 2, we show how to reduce observational equivalence to symbolic equivalence of constraint systems (Section 6.2). For a large class of processes, this allows us to conclude by using an existing decision procedure by M. Baudet [Bau05, Bau07]. In Section 6.3, we propose an alternative procedure that has been implemented and that works well in practice.

6.1 Going beyond with the ProVerif tool

We develop a formal verification technique for proving observational equivalence of cryptographic protocols. We focus on proving observational equivalence between processes P and Q having the same structure and differing only in the choice of terms. In Section 4.4, we have seen that the ProVerif tool makes some progress in this direction. However, the method developed for proving observational equivalence is not complete and is unable to prove certain interesting properties as those presented in Section 4.3.

In [27], we have expanded the scope of ProVerif, to provide reasoning about further equivalences. We have developed an algorithm to enable automated reasoning. Using this approach, we provided the first automated proof that the electronic voting protocol by Fujioka, Okamoto, and Ohta [FOO92b] satisfies privacy. As a second case study, we provide a formal proof that the Direct Anonymous Attestation [BCC04] protocol also satisfies privacy. We also use this technique to establish privacy properties in the context of vehicular ad hoc network.

6.1.1 Some extensions

To overcome the limitations presented in Section 4.4.2 that prevent us to analyse several protocols using ProVerif, we have extended the applied pi calculus with strong phases and data swapping.

Strong phases. Many protocols can be broken into phases, and their security properties can be formulated in terms of these phases. Typically, for instance, if a protocol discloses a session key after the conclusion of a session, then the secrecy of the data exchanged during the session may be compromised but not its authenticity. To enable modelling of protocols with several phases, the syntax of processes is supplemented with a phase prefix "$\texttt{phase}\ t; P$". Intuitively t represents a global clock, and the process "$\texttt{phase}\ t; P$" is active only during phase t.

However, in order to model electronic voting protocols, we need to consider global synchronisation. We have to express that the registration is closed before the voting phase starts. To achieve this, the syntax of processes is supplemented with a strong phase prefix "$\texttt{strong phase}\ t; P$". A strong phase represents a global synchronisation and t represents the global clock. The process $\texttt{strong phase}\ t; P$ is active only during strong phase t and a strong phase progression may only occur once all the instructions under the previous phase have been executed.

Example 6.1 *The process* $\texttt{strong phase}\ 1; \mathit{out}(c, a) \mid \texttt{strong phase}\ 2; \mathit{out}(c, b)$ *cannot output b without having previously output a. Note that this behaviour is possible in presence of a "weak" phase.*

Data swapping. The equivalence $\mathsf{out}(c, a) \mid \mathsf{out}(c, b) \approx_\ell \mathsf{out}(c, b) \mid \mathsf{out}(c, a)$ holds trivially since the processes are in fact structurally equivalent. But the corresponding bi-process $P = \mathsf{out}(c, \mathrm{choice}[a, b]) \mid \mathsf{out}(c, \mathrm{choice}[b, a])$ does not satisfy diff-equivalence and therefore the equivalence cannot be proved by ProVerif.

Actually, we have that $\mathsf{fst}(P) = \mathsf{out}(c, a) \mid \mathsf{out}(c, b)$ and $\mathsf{snd}(P) = \mathsf{out}(c, b) \mid \mathsf{out}(c, a)$. Since $\mathsf{out}(c, b) \mid \mathsf{out}(c, a) \equiv \mathsf{out}(c, a) \mid \mathsf{out}(c, b)$ it seems reasonable to rewrite the process $\mathsf{snd}(P)$ as $\mathsf{out}(c, a) \mid \mathsf{out}(c, b)$, enabling us to write P as $\mathsf{out}(c, \mathrm{choice}[a, a]) \mid \mathsf{out}(c, \mathrm{choice}[b, b])$. Our

new bi-process satisfies diff-equivalence, and thus observational equivalence. It therefore seems possible (under certain circumstances) to *swap* values from the left to the right side of the parallel operator. Sometimes the swap is not done initially but instead immediately after a strong phase. In such a case, we have to ensure that the data used during the current phase (nonces, inputs) are passed to the next phase. This can be done by using private channels. To specify data swapping we introduce the special comment *(**swap*)* in process descriptions, which can be seen as a *proof hint*.

Our translator. To allow automated reasoning we have proposed a translator which accepts as input processes written in our extended language. It will also include a single main process and sub-processes of the form "`let` $P = Q$", subject to some restrictions. The translator outputs processes in the standard language of ProVerif, which can be automatically reasoned about by the software tool. If ProVerif is now able to establish equivalence of the resulting bi-process, this means that the bi-process P given in input is such that $\mathsf{fst}(P) \approx_\ell \mathsf{snd}(P)$.

Recently, the transformation has been revisited [SB10] and implemented in the ProSwapper tool [Smy10]:

`http://www.bensmyth.com/proswapper.php`.

6.1.2 Applications

In this section, we demonstrate the usefulness of our translator on several case studies that are fully described in [27, 15].

Electronic voting protocol by Fujioka, Okamoto, and Ohta (FOO). The protocol involves voters, an administrator and a collector and it is based on blind signatures and bit commitments. The administrator is responsible for verifying that only eligible voters can cast votes and the collector handles the collecting and publishing of votes. The protocol requires three strong phases (registration, voting, tallying). The separation of the protocol into strong phases is crucial for privacy to hold.

In [6], we rely on hand proof techniques to show privacy on FOO: ProVerif is unable to prove it directly. Our modelling of FOO is similar to the one we have proposed in [6]. In addition, we have to provide a data swapping hint to allow our translator to produce an output suitable for automatic verification using ProVerif.

We use our translator to remove all instances of strong phases and handle data swapping. Our translator produces a process, which is suitable for automatic verification using ProVerif. Hence, using this approach, we provided the first automated proof that the FOO protocol satisfies privacy.

Direct Anonymous Attestation (DAA) protocol. This protocol provides a means for remotely authenticating a trusted platform whilst preserving the user's privacy [BCC04]. In [SRC07], B. Smyth *et al.* have shown that corrupt administrators are able to violate the privacy of the host. Using our extended calculus we are able to provide a formal and automatic proof that the rectified protocol proposed in [SRC07] satisfies its privacy requirements.

The protocol can be seen as a group signature scheme without the ability to revoke anonymity and an additional mechanism to detect rogue members. In broad terms the *host*

contacts an *issuer* and requests membership to a group. If the issuer wishes to accept the request, it grants the host/TPM an *attestation identity credential*. The host is now able to anonymously authenticate itself as a group member to a *verifier* with respect to its credential. The protocol is initiated when a host wishes to obtain a credential. This is known as the join protocol. Once the host has obtained an anonymous attestation credential from the issuer it is able to produce a signature proof of knowledge of attestation on a message m. This is known as the sign/verify protocol. The DAA protocol makes extensive use of signature proofs of knowledge (SPK) to prove knowledge of and relations among discrete logarithms.

Intuitively, the DAA protocol satisfies privacy whenever a process where Alice interacts with the verifier is observationally equivalent to when Bob interacts with the verifier. For privacy we require that both Alice and Bob have completed the join protocol. This can be formally stated as an observational equivalence between two processes. We use the strong phase and data swapping commands introduced by our extension to the calculus to ensure synchronisation. The two instances of the DAA processes must first execute all instructions of DAAJoin before moving onto DAASign. The separation of the protocol into strong phases is crucial for privacy to hold.

We use our translator to remove all instances of strong phases from our encoding and produce code suitable for input to ProVerif. Our translator produces a process which permits the automatic verification of the privacy property using ProVerif. We are able to detect the vulnerability in the original DAA protocol and to prove the optimisation presented in [SRC07].

CMIX protocol in vehicular ad hoc network. The CMIX protocol [FRF+07] distributes keys for encrypting beacon messages while in the mix-zone with the goal of preventing an attacker from linking the pseudonym of an in-coming vehicle with the pseudonym it uses when leaving. Every vehicle is equipped with a tamper resistant device (TRD) allowing access to its contents only through its API. Every vehicle has a fresh non-empty set of these key-pseudonym pairs stored in its TRD. One pair is marked as current, to be used when sending messages. The zone key is then used to encrypt and decrypt beacon messages while inside the mix-zone.

We performed our analysis in two models: an ideal model where the vehicles are assumed to know the mix-zone encryption key and a CMIX model where this key is distributed using the CMIX protocol. From our ideal model analysis, we extract a set of scenarios where it is possible for a 'perfect' key distribution protocol to guarantee privacy.

We then evaluate the CMIX protocol with respect to these scenarios. We consider all scenarios where privacy is provable in the ideal model. First, we add one session of the CMIX protocol to both vehicle processes. We found that in all cases where privacy was possible in the ideal model, it was also possible here. We use our translator to obtain these results. Actually, if ProVerif is able to establish observational equivalence on the process outputted by our translator, this means that the equivalence really holds. In case ProVerif is not able to conclude, we have to check that this corresponds to a real attack. This was indeed the case in all our experiments. In particular, we report some scenarios in which the use of the CMIX protocol can prevent privacy from being achieved when a second session of the CMIX protocol is triggered before the vehicle left the mix-zone.

6.2 From observational equivalence to symbolic equivalence

The aim of this section is to provide some reduction results. More precisely, we show that observational equivalence can be reduced to the problem of checking symbolic equivalences of pairs of constraint systems. We can then conclude by using the decision procedure proposed in [Bau05, Bau07] for the class of subterm convergent equational theories. We have also developed our own procedure that is presented in Section 6.3.

6.2.1 Symbolic equivalence of pair of constraint systems

The following definition of constraint system is consistent with the definition given in Chapter 2. However, we need to generalize the usual definition. Indeed, the constraint solving method we presented in Chapter 2 is complete only *w.r.t.* what an attacker can deduce, but not *w.r.t.* how it can be deduced. If an attacker has different ways to deduce a given message in two different experiments, he could distinguish between them. This is the main purpose of the variables X_i: they are used to record the recipe that has been used to deduce s_i. This information is needed to capture observational equivalence.

An *initial deducibility constraint* is either $\bot$ or consists of:

1. a frame $\phi = \{ax_1 \rhd u_1, \ldots, ax_m \rhd u_m\}$, whose size is some m (terms $u_1, \ldots, u_m$ may contain variables);
2. a sequence $D = X_1, i_1 \stackrel{?}{\vdash} s_1;\ \ldots;\ X_n, i_n \stackrel{?}{\vdash} s_n$ where
   - $X_1, \ldots, X_n$ are distinct variables, $s_1, \ldots, s_n$ are terms, and $0 \leq i_1 \leq \ldots \leq i_n \leq m$.
   - for every $0 \leq k \leq m$, $var(u_k) \subseteq \bigcup_{i_j < k} var(s_j)$;
3. a conjunction E of equations between terms.

The variables X_i represent the recipes that might be used to deduce the right-hand side of the deducibility constraint. The indices indicate which initial segment of the frame can be used. An E-*solution* of an initial deducibility constraint $\mathcal{C} = (\phi, D, E)$ consists of

- a substitution σ from $var(\{s_1, \ldots, s_n\})$ to ground terms, and
- a substitution θ mapping $X_1, \ldots, X_n$ to ground recipes, *i.e.* terms built from public function symbols, and special variables $ax_1, \ldots, ax_m$,

such that:

- for every $X_i, j \stackrel{?}{\vdash} s_i$ in D, $var(X_i\theta) \subseteq \{ax_1, \ldots, ax_j\}$ and $X_i\theta(\phi\sigma) =_{\mathsf{E}} s_i\sigma$;
- for every equation $u \stackrel{?}{=} v$ in E, we have that $u\sigma =_{\mathsf{E}} v\sigma$.

We denote by $Sol_{\mathsf{E}}(\mathcal{C})$ (or simply $Sol(\mathcal{C})$ when E is clear from the context) the set of E-solutions of $\mathcal{C}$. By convention, we have that $Sol_{\mathsf{E}}(\bot) = \emptyset$.

Intuitively, we want to check static equivalence on any possible trace. More precisely, we want that any experiment θ performed by the attacker on $\mathcal{C}$ has a counterpart on $\mathcal{C}'$ such that the resulting sequence of messages that are observed on both sides are statically equivalent. This is captured by the following definition:

Definition 6.1 (symbolic equivalence) *Let $\mathcal{C}$ and $\mathcal{C}'$ be two constraint systems whose corresponding frames are ϕ and ϕ'. The constraint system $\mathcal{C}$ is* symbolically equivalent *to $\mathcal{C}'$, denoted $\mathcal{C} \approx_s \mathcal{C}'$, if:*

- *for all $(\theta, \sigma) \in Sol(\mathcal{C})$, there exists σ' such that $(\theta, \sigma') \in Sol(\mathcal{C}')$, and $\phi\sigma \sim \phi'\sigma'$,*
- *for all $(\theta, \sigma') \in Sol(\mathcal{C}')$, there exists σ such that $(\theta, \sigma) \in Sol(\mathcal{C})$, and $\phi\sigma \sim \phi'\sigma'$.*

In the following section, we will see how equivalence between simple processes can be expressed via symbolic equivalence of *initial pairs of constraints*, that is a pair of the form (ϕ_1, D_1, E_1), (ϕ_2, D_2, E_2) where both initial constraint systems have the same shape. More precisely, we have that:

- $\phi_1 = \{ax_1 \triangleright u_1, \ldots, ax_m \triangleright u_m\}$, and $D_1 = X_1, i_1 \stackrel{?}{\vdash} s_1;\ \ldots;\ X_n, i_n \stackrel{?}{\vdash} s_n$;
- $\phi_2 = \{ax_1 \triangleright v_1, \ldots, ax_m \triangleright v_m\}$, and $D_2 = X_1, i_1 \stackrel{?}{\vdash} t_1;\ \ldots;\ X_n, i_n \stackrel{?}{\vdash} t_n$.

Or else it is a pair as above, in which one of the components is replaced with $\bot$.

6.2.2 The case of simple processes

We consider any signature and equational theory. However, we do not consider the full applied pi calculus but only a restricted fragment. For example, it is generally assumed that all communications are controlled by the attacker thus private channels between processes are not accurate (they should rather be implemented using cryptography). In addition, the attacker schedules the communications between processes thus he knows exactly to whom he is sending messages and from whom he is listening. Thus we assume that each process communicates on a personal channel.

More precisely, we consider the fragment of *simple processes* built on *basic processes* (see [21] for a formal definition of simple processes). A basic process represents a session of a protocol role where a party waits for a message of a certain form or checks some equalities and outputs a message accordingly. Then the party waits for another message or checks for other equalities and so on. Intuitively, any protocol whose roles have a deterministic behavior can be modeled as a simple process. Most of the roles are indeed deterministic since an agent should usually exactly know what to do once he has received a message. In particular, all protocols of the J. Clark and J. Jacob library [CJ97] can be modelled as simple processes. However, it is interesting to notice that protocols with deterministic behavior are usually not modelled within our fragment since a single channel is used for all communications. We think however that using a single channel does not provide enough information to the attacker since he is not able to schedule exactly the messages to the processes and he does not know from which process a message comes from while this information is usually available (via *e.g.* IP adresses). For example, a role emitting the constant a twice would be modelled by $P_1 = \mathtt{out}(c, a).\mathtt{out}(c, a)$ while two roles emitting each the constant a would be modeled by $P_2 = \mathtt{out}(c, a) \mid \mathtt{out}(c, a)$. Then P_1 and P_2 are observationally equivalent while the two protocols could be distinguished in practice, which is reflected in our modelling in simple processes.

In [21], we have shown that simple processes are determinate (see Definition 4.4). Indeed, since each basic process has its own channel to send and receive messages, all the communications are visible to the attacker. Moreover, the attacker knows exactly who is sending a message or from whom he is receiving a message.

Proposition 6.1 *[21] Any simple process is determinate.*

Hence, relying on Theorem 4.1, to decide observational equivalence for simple processes, it remains to show how to decide trace equivalence. First, we can show that trace equivalence is exactly captured by a notion of symbolic trace equivalence. Moreover, since for simple processes without replication, there are a finite number of symbolic traces, we have the following result.

Theorem 6.1 *[21] Let* E *be a subterm convergent equational theory. Let* A *and* B *be two simple processes without else branch nor replication. The problem whether* A *and* B *are observationally equivalent is co-NP-complete.*

The decidability of observational equivalence follows from the fact that symbolic equivalence of (positive) constraint systems is decidable for subterm convergent equational theories [Bau05, CR11]. The NP-TIME decision procedure for non-observational equivalence works as follows:

- Guess a symbolic (annotated) trace tr;
- Compute (in polynomial time) the corresponding initial constraint system $\mathcal{C}$ (resp. $\mathcal{C}'$) associated to A (resp. B);
- Check whether $\mathcal{C}$ and $\mathcal{C}'$ are in symbolic equivalence.

Due to a result first proved by M. Baudet [Bau05], we know that the last step can be done in NP-TIME for convergent subterm theories thus we deduce that the overall procedure is NP-TIME. NP-hardness is obtained using the usual encoding [RT03]. Recently, M. Rusinowitch and Y. Chevalier [CR11] proposed another decision procedure based on an extension of the small attack property. They show that, if two processes are not equivalent, then there must exist a small witness of non-equivalence. A decision of equivalence can be derived by checking every possible small witness.

6.2.3 The case of general processes

The class of simple processes appears to be sufficient to represent most of the cryptographic protocols. However, even if private and/or anonymous channels are not accurate, they are useful for modelling purposes. Hence, it may be interesting to go beyond this class of simple processes. In [28, 5], relying on constraint systems, we have proposed a symbolic semantics for the full applied pi without replication. As in the previous approaches, by treating inputs symbolically, our semantics avoids potentially infinite branching of execution trees due to the inputs from the environment.

The semantics of the applied pi calculus is not well-suited for defining such a symbolic semantics. In particular, defining a symbolic structural equivalence which is both sound and complete seems impossible. The absence of sound and complete symbolic structural equivalence significantly complicates the proofs showing that the symbolic semantics we proposed

is sound and complete *w.r.t.* the concrete one. This lead us to define a more restricted semantics which will provide an intermediate representation of applied pi calculus processes. These 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. This calculus is of independent interest. Actually, we have partly reused this intermediate calculus to obtain the result presented in Section 6.2.2. This intermediate calculus has also been reused by J. Liu and H. Lin [LL10] to propose a complete symbolic bisimulation in presence of replication.

Then, we have proposed a definition of symbolic labelled bisimilarity based on the notion of symbolic equivalence of constraint systems. Note that since the processes we consider are not simple, we can not assume that one symbolic move of a process can be mimicked by a single symbolic move in the other process. Because of this, our technique suffers from several sources of incompleteness. For instance, we are not able to establish that the two following processes are bisimilar.

$$\begin{aligned} P_1 &= \mathtt{in}(c,x).\mathtt{out}(c,a) \\ Q_1 &= \mathtt{in}(c,x).\text{if } x = a \text{ then } \mathtt{out}(c,a) \text{ else } \mathtt{out}(c,a) \end{aligned}$$

In our setting, the instantiation of input variables is postponed until the point at which they are actually used. This allows one to not decide the instantiation of the variable when the input has to be performed but later on. This instantiation may depend on the choice that has been made to reach that point. Hence, this prevents us to establish that the two following processes are bisimilar.

$$\begin{aligned} P_2 &= \nu c_1.\mathtt{in}(c_2,x).(\mathtt{out}(c_1,b) \mid \mathtt{in}(c_1,y) \mid \text{if } x = a \text{ then } \mathtt{in}(c_1,z).\mathtt{out}(c_2,a)) \\ Q_2 &= \nu c_1.\mathtt{in}(c_2,x).(\mathtt{out}(c_1,b) \mid \mathtt{in}(c_1,y) \mid \mathtt{in}(c_1,z).\text{if } x = a \text{ then } \mathtt{out}(c_2,a)) \end{aligned}$$

Although our symbolic bisimulation is not complete, as shown above, we are able to prove observational equivalence on interesting examples. Actually, our symbolic bisimulation is sufficiently complete to deal with examples of anonymity properties (vote-privacy, receipt-freeness, ...) arising in protocol analysis. Again, we can directly build on existing works [Bau05, CR11] and obtain a decision procedure for our symbolic bisimulation for the class of subterm convergent equational theories and processes that do not contain else branches.

Related work. Recently, a relatively simple symbolic transition system and bisimulation equivalence that is fully abstract *w.r.t.* concrete bisimulation has been proposed for psi-calculi [BJPV09, JVP10]. Psi-calculi can be more general than other proposed extensions of the pi-calculus such as the applied pi calculus that we consider here. In psi-calculi, they are helped significantly by the absence of structural equivalence rules which are rather complex in the applied pi calculus.

6.3 Decision procedure for symbolic equivalence

In [17], we have proposed a procedure to decide symbolic equivalence. We consider pairing, signature, symmetric and asymmetric encryptions only. Our main goal was to provide a

procedure together with an efficient implementation. The main result stated in [17] is a decision procedure for symbolic equivalence of an initial pair of constraint systems.

Theorem 6.2 *[17] Given an initial pair $(\mathcal{C},\mathcal{C}')$, it is decidable whether $\mathcal{C} \approx_s \mathcal{C}'$.*

This result in itself is already known (*e.g.* [Bau05, CR11]), but the known algorithms cannot yield any reasonable implementation. Even, the recent algorithm proposed by A. Tiu and J. Dawson [TD10] has not been implemented and should probably be improved to lead to a reasonable implementation. Similarly, in [CR11], they show that, if two processes are not equivalent, then there must exist a small witness of non-equivalence. However, the number of small witnesses is very large as all terms of size smaller than a given bound have to be considered. Consequently, neither this method nor the previous one have been implemented.

6.3.1 Our algorithm in a nutshell

Our decision algorithm works by rewriting pairs of constraint systems that have been extended to keep track of some information (for instance, we consider equations between recipes), until a trivial failure or a trivial success is found. These rules are branching: they rewrite a pair of constraint systems into two pairs of constraint systems. Transforming the pairs of constraints therefore builds a binary tree. Termination requires to keep track of some information, that is recorded using a set F of flags attached to each deducibility constraint (*e.g.* $\mathtt{NoCons}_f,\dots$). The flags are additional constraints that restrict the recipes. The complete set of transformation rules can be found in [17]. Below, we only present some instances of those rules to explain how the algorithm works. They are displayed for a single constraint system only.

Rule $\textsc{Cons}_{\mathsf{senc}}$:

$$X, i \overset{?}{\vdash}_F \mathsf{senc}(t_1,t_2) \;\rightarrow\; \begin{cases} X_1, i \overset{?}{\vdash}_F t_1;\;\; X_2, i \overset{?}{\vdash}_F t_2;\;\; X \overset{?}{=} \mathsf{senc}(X_1,X_2) \\ X, i \overset{?}{\vdash}_{F+\mathtt{NoCons}_{\mathsf{senc}}} \mathsf{senc}(t_1,t_2) \end{cases}$$

If $\mathtt{NoCons}_{\mathsf{senc}} \notin F$ and X_1, X_2 are fresh variables.
This rule simply guesses whether the top symbol of the recipe used to deduce $\mathsf{senc}(t_1,t_2)$ is senc or not. Either it is, and then we can split the constraint, or it is not and we add a flag forbidding this.

Rule Eq-left-left:

$$\zeta_1 \triangleright_{F_1} u_1;\;\; \zeta_2 \triangleright_{F_2} u_2 \;\rightarrow\; \begin{cases} \zeta_1 \triangleright_{F_1} u_1;\;\; \zeta_2 \triangleright_{F_2} u_1;\;\; u_1 \overset{?}{=} u_2 \\ \zeta_1 \triangleright_{F_1} u_1;\;\; \zeta_2 \triangleright_{F_2} u_2;\;\; u_1 \overset{?}{\neq} u_2 \end{cases}$$

This rule covers the comparisons that an attacker could perform at various stages. This equality rule guesses equalities between terms known by the attacker.

Rules above are displayed for a single constraint system only. We explain here how they are extended to pairs of constraint systems. If one of the constraint systems is $\bot$, then we proceed as if there was a single constraint. Otherwise, the indices i and the recipes X, ζ (resp. $X_1, X_2, \zeta_1, \zeta_2$) matching the left side of the rules *must be identical in both constraints*: we apply the rules at the same positions in both constraint systems. We have to explain now what happens when, on a given pair $(\mathcal{C}, \mathcal{C}')$ a rule can be applied on $\mathcal{C}$ and not on $\mathcal{C}'$ (or the converse). Therefore, the rule $\textsc{Cons}_{\mathsf{senc}}$, when applied to pairs of constraint systems comes in three versions: either the rule is applied on both sides or, if $X, i \overset{?}{\vdash} \mathsf{senc}(t_1, t_2)$ is in $\mathcal{C}$, and $X, i \overset{?}{\vdash} x$ is in $\mathcal{C}'$, we may apply the rule on the pair of constraint systems, adding to $\mathcal{C}'$ the equation $x \overset{?}{=} \mathsf{senc}(x_1, x_2)$ where x_1 and x_2 are fresh variables. The third version is obtained by switching $\mathcal{C}$ and $\mathcal{C}'$. Note that this may introduce new variables, that yield a termination issue. For the rules Eq-left-left, we require that at least one new non-trivial equality (or disequality) is added to one of the two constraint systems (otherwise there is a trivial loop). For all rules, if a rule is applicable on one constraint and not the other, we do perform the transformation, however replacing a constraint with $\bot$ when a condition becomes false.

Our algorithm can be stated as follows:

- Construct, from an initial pair of constraint systems $(\mathcal{C}_0, \mathcal{C}'_0)$ a tree, by applying as long as possible a transformation rule to a leaf of the tree.
- If, at some point, there is a leaf to which no rule is applicable and that is labeled $(\mathcal{C}, \bot)$ or $(\bot, \mathcal{C})$ where $\mathcal{C} \neq \bot$, then we stop with $\mathcal{C}_0 \not\approx_s \mathcal{C}'_0$.
- Otherwise, if the construction of the tree stops without reaching such a failure, return $\mathcal{C}_0 \approx_s \mathcal{C}'_0$.

Our algorithm can also be used to decide static equivalence of frames, as well as the (un)satisfiability of a constraint system (this can be used to decide reachability properties – see Chapter 2). Furthermore, in case of failure, a witness of the failure can be returned, using the equations of the non-$\bot$ constraint.

Termination requires to keep track of some information, that is recorded using flags. In general the rules might not terminate. Fortunately, there is however a simple and complete strategy that avoids the non-terminating behaviours. Our transformation rules are sound: if all leaves are success leaves, then the original pair of constraint systems is equivalent. They are finally complete: if the two original constraint systems are equivalent then any pairs of constraint systems resulting from a rewriting steps are also equivalent.

6.3.2 Implementation: the ADECS tool

An Ocaml implementation of the procedure described in [17], as well as several examples, are available at `http://www.lsv.ens-cachan.fr/~cheval/programs/index.php` (around 5000 lines of Ocaml). The implementation closely follows the transformation rules that we described. For efficiency reasons, a strategy on checking the rules applicability has been designed in addition.

We checked the implementation on examples of static equivalence problems, on examples of satisfiability problems, and on symbolic equivalence problems that come from classical protocols such as the Needham-Schroeder protocol [NS78], the hanshake protocol [GLNS93],... On

all examples the tool terminates in less than a second (on a standard laptop). Note that the input of the algorithm is an initial pair of constraints: checking the equivalence of protocols would require in addition an interleaving step, that could be expensive.

We cannot really compare with other tools (there are no such tools); we simply tested the program on some home made benchmarks as well as on some constraint systems derived from well-known protocols from the literature.

6.4 Perspectives

The transformation we proposed in Section 6.1 to expand the scope of ProVerif is quite promising. It is simple and the suitability of our translator is demonstrated by analysing several case studies coming from different applications such as e-voting and vehicular ad hoc networks. Actually, it remains room to improve the ProVerif tool. One possible direction will be to extend the ProVerif tool in order to be able to deal with more equational theories, such as exclusive-or or trapdoor commitment. A way to achieve this would be to exploit the finite variant property that has been introduced in [38] in order to get rid of some algebraic properties. R. Küsters and T. Truderung used this technique in [KT11]. In particular, they analyse protocols relying on exclusive-or using the ProVerif tool. To achieve this, they have to get rid of the algebraic properties of the exclusive-or operator since ProVerif is not able to deal with these properties. However, they only consider trace security properties (secrecy and authentication). Their result does not apply for equivalence-based property. The exclusive-or operator will be particularly useful to analyse RFID protocols [BCdH10] for which privacy-type properties such as untraceability [ACRR10, BCdH10] play an important role.

Despite the extensions described above, there will remain situations where ProVerif will not be able to conclude. For instance, this is typically what happens in the case study described in [ACRR10]. They study the e-passport protocol and they report an attack on the French version. They also show that other nations, like UK, Germany, Ireland, and Russia have avoided this linkability attack. However, ProVerif is unable to prove the desired equivalence property since the two processes involved in this equivalence are not in diff-equivalence and it seems difficult to come with a transformation or an encoding allowing us to go beyond this situation. Actually, the other methods presented in this chapter will not allow us to conclude on this case study that uses a conditional with an else branch. However, it seems possible to extend these techniques to overcome this limitation. As future work, it would be interesting to extend the class of processes we considered in different ways. For example, we would like to extend our decision result to else branches. This would require adding disequality tests in set of constraints and adapting the procedure for deciding symbolic equivalence of constraint systems accordingly. This will still not be sufficient to deal with the e-passport case study. In order to be able to establish the equivalence, it is necessary to extend symbolic equivalence of pairs of constraint systems to pairs of sets of constraint systems. We are currently investigating these different aspects. Lastly, in order to get an automatic tool for deciding equivalence of processes, we have to move from symbolic equivalence between constraint systems (or sets of constraint systems) to equivalence between processes. This requires computing all interleavings of actions, a step which could be prohibitive from the computation time point of view. Hence, in order to get an efficient procedure, it is necessary to come with some optimisations in order to reduce the search space and the number of

interleavings. This problem is not specific to equivalence-based properties and has already been studied in the context of trace-based properties [CJM03, MVB10]. However, discarding some "symbolic" interleavings appears to be challenging for equivalence-based properties such as those studied in this chapter.

Part III

Composition results

Chapter 7

Parallel composition

Contents

SYMBOLIC techniques showed to be a very useful approach for the modelling and analysis of security protocols: they improved our understanding of security protocols, allowed discovering flaws [Low96], and provided support for protocol design [DDMP04]. These techniques also resulted in the creation of powerful automated analysis tools (*e.g.* [Bla01, BMV05, ABB+05, Cre08b]), and impacted on several protocol standards used every day, *e.g.* [CJS+08]. However, these analyses usually consider that the protocol is executed in isolation, ignoring other protocols that may be executed in parallel. The assumption that another parallel protocol cannot interfere with the protocol under investigation is valid if the two protocols do not share any secret data (such as cryptographic keys or passwords). But if such data is shared between protocols, then this assumption is not valid.

Under certain conditions, we may have that

$$\text{if } P_1 \text{ and } P_2 \text{ are secure then } P_1 \mid P_2 \text{ is secure.}$$

For example, in the context of applied pi calculus [AF01]), "is secure" is often formalised as observational equivalence to some specification. We have that $P_1 \approx S_1$ and $P_2 \approx S_2$ imply $P_1 \mid P_2 \approx S_1 \mid S_2$, where S_1 and S_2 are specifications, and therefore the security of the composition follows from the security of each protocol. Here, the compositionality of

security relies on two facts. First, as mentioned, security means observational equivalence to a specification; the attacker is an *arbitrary context*, and $P_1 \approx S_1$ means P_1 and S_1 are equivalent in any environment. Second, by forming the composition $P_1 \mid P_2$ we have made the assumption that P_1 and P_2 do not share any secret.

Now suppose that P_1 and P_2 do share a secret key k. To prove that their security composes, one would like to show that

$$\text{if } \texttt{new}\ k.P_1 \text{ and } \texttt{new}\ k.P_2 \text{ are secure then } \texttt{new}\ k.(P_1 \mid P_2) \text{ is secure.}$$

Note in particular that $\texttt{new}\ k.(P_1 \mid P_2)$ is different from $(\texttt{new}\ k.P_1) \mid (\texttt{new}\ k.P_2)$ because the later refers to two different secrets as they have different scope. In contrast with the previously mentioned composition result, this one does not hold in general.

In this chapter, we consider this notion of parallel composition under long-term keys (or passwords). First, we propose a result that allows one to safely compose different protocols that share some long-term keys (see Section 7.1). Unfortunately, this transformation relies on a static tag and can not be applied to compose sessions coming from the same protocol. Hence, in Section 7.2, we provide a more involved transformation that allows us to compose sessions. In both results, we consider trace security properties such as secrecy and authentication. Lastly, we investigate the case of password-based protocols. In particular, we provide a syntactic condition to ensure that a user can use the same password for different protocols without compromising the security of his password. This also relies on a tagging scheme to avoid interaction between messages coming from different protocols. This result is described in Section 7.3.

7.1 Composing different protocols sharing keys

Consider for example the two following naive protocols.

$$P_1 : \quad A \to B : \quad \mathsf{aenc}(s, \mathsf{pk}(B)) \qquad\qquad P_2 : \quad \begin{array}{ll} A \to B : & \mathsf{aenc}(N_a, \mathsf{pk}(B)) \\ B \to A : & N_a \end{array}$$

In protocol P_1, the agent A simply sends a secret s encrypted under B's public key. In protocol P_2, the agent sends some fresh nonce to B encrypted under B's public key. The agent B acknowledges A's message by forwarding A's nonce. While P_1 executed alone easily guarantees the secrecy of s, even against an active adversary, the secrecy of s is no more guaranteed when the protocol P_2 is executed. Indeed, an adversary may use the protocol P_2 as an oracle to decrypt any message. More realistic examples illustrating interactions between protocols can be found in *e.g.* [KSW97].

7.1.1 Main result

In this section, we first introduce and discuss the hypotheses we need to safely compose protocols. This result is only stated informally. A formal statement together with its detailed proofs can be found in [29, 7].

Disjoint encryption. To avoid a ciphertext from a protocol P_1 to be decrypted in another protocol P_2, we consider protocols that satisfies *disjoint encryption*, *i.e.* any two encrypted sub-messages coming from two different protocol specifications cannot be unified. However, protocols that use common keys (*e.g.* common public keys) may not enjoy the disjoint encryption property. A way to force disjoint encryption is to use tags. Requiring that two protocols satisfy disjoint encryption can be very easily achieved in practice: it is sufficient for example to add the name of the protocol in each encrypted term.

Continuing our example, let us consider the two slightly modified protocols.

$$P_1' : \quad A \to B : \quad \mathsf{aenc}(\langle 1, s\rangle, \mathsf{pk}(B)) \qquad\qquad P_2' : \quad \begin{array}{ll} A \to B : & \mathsf{aenc}(\langle 2, N_a\rangle, \mathsf{pk}(B)) \\ B \to A : & N_a \end{array}$$

Our main composition theorem will ensure that P_1' can be safely executed together with P_2', without compromising the secrecy of s.

Critical long-term keys. Disjoint encryption is not a sufficient condition. Indeed critical long-term keys should not be revealed in clear. Consider for example the following two protocols. Note that they satisfy disjoint encryption since P_4 has no encrypted subterm.

$$P_3 : \quad A \to B : \quad \mathsf{aenc}(\langle 1, s\rangle, k_{ab}) \qquad\qquad P_4 : \quad A \to B : \quad k_{ab}$$

The security of protocol P_3 is compromised by the execution of P_4. Thus we will require that long-term keys (except possibly the public ones) do not occur in plaintext in the protocol. This is not a real restriction since not disclosing the long term private keys in plaintext (even under encryption) corresponds to a prudent practice.

Main result. We show that two protocols can be safely composed as soon as they satisfy the disjoint encryption assumption and that critical long-term keys do not appear in plaintext. Our result works for protocols that are built on pairing, symmetric and asymmetric encryptions, signatures, and hash functions. We prove this composition result for a class of security properties that includes secrecy and different kinds of authentication (*e.g.* aliveness, agreement).

To establish this result, we rely on constraint systems (see Chapter 2). We prove our composition result by contradiction and we first show that messages from two combined protocols do not need to be mixed up to mount an attack. For this purpose, we use the simplification rules presented in Section 2.1.3. This set of rules allow us to control the form of the execution traces.

Related work. A result closely related to ours is the one of Guttman and Thayer [GT00]. They show that two protocols can be safely executed together without damaging interactions, as soon as the protocols are "independent". The independence hypothesis requires in particular that the set of encrypted messages that the two protocols handle should be different. As in our case, this can be ensured by giving each protocol a distinguishing value that should be included in the set of encrypted messages that the protocol handles. However, the major difference with our result is that this hypothesis has to hold not only on the protocol *specification* but also on any valid *execution* of the protocol. In particular, considering again the protocol P_2', an agent should not accept a message of the form $\mathsf{aenc}(\langle 2, \mathsf{senc}(\langle 1, m\rangle, k), \mathsf{pk}(B))$

while he might not be able to decrypt the inside encryption and detect that it contains the wrong identifier.

Another result has been obtained by Andova *et al.* for a broader class of composition operations [ACS+08]. Their result does not allow one to conclude when no typing hypothesis is assumed (that is, when agents are not required to check the type of each component of a message) or for protocols with ciphertext forwarding, that is, when agents have to forward unknown message components. Recently, V. Cortier and Ş. Ciobâcă have extended the result presented in this section in different ways [CC10]. First, they consider a broader class of cryptographic primitives. Second, their result allows one to consider various ways of combining protocols such as sequentially or in parallel, possibly with inner replications. They have to assume that protocols do not share any cryptographic primitives unless they are tagged. This means that their result does not allow one to compose protocols that rely on pairs without tagging them.

7.1.2 Applications

Security protocols can be analyzed using several existing tools, *e.g.* [Bla01, ABB+05]. The security of a protocol P is however guaranteed provided that no other protocols share any of the private data of P. Our result shows that, once the security of an isolated protocol has been established, this protocol can be safely executed in an environment that may use some common data provided disjoint encryption is satisfied (and that long term private keys are not sent in plaintext). This condition is easy to check but might not be satisfied by existing protocols. A simple way to ensure it is to add the name of the protocol each time a party performs an encryption.

For security reasons however, most protocols actually make use of different keys. Here, we provide a simple criteria for safely composing protocols that share keys. This would allow to save both memory (for storing keys) and time since generating keys is time consuming in particular in the case of public key encryption. For example, the SSL protocol should contain the name "ssl" in any of its encrypted messages. This would ensure that no harmful interaction can occur with any other protocols even if they share some data with the SSL protocol, provided that these other protocols are also tagged. In other words, to avoid harmful interaction between protocols, one should simply use a tagged version of them.

There are also situations were running different protocols with common keys already occur. We provide three examples of such cases.

- It is often the case that several versions of a protocol can be used concurrently. In this case and for backward compatibility reasons, the same keys can be used in the different versions of the protocol, which may lead to potentially dangerous interactions.

- When encrypting emails the same public key can be used for two distinct encryption protocols: the PGP protocol (Pretty Good Privacy) and its open source version OpenPGP. Note that the PGP protocol contains also other sub-protocols such as digitally signing message, all sharing the same public key infrastructure.

- In the slightly different context of APIs, J. Clulow [Clu03b] discovered an attack when the VISA PIN verification values (PVV) protocol and the IBM CCA API share the same verification key.

7.2 Composing sessions of the same protocol

An obvious situation where we have to do composition in presence of shared long-term keys is when we want to compose sessions coming from the same protocol. However, in such a situation, the transformation proposed in Section 7.1 that consists of adding a *static tag* in each encrypted subterm can not be applied.

In [24], we have proposed a more involved 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. To achieve this, we rely on dynamic tags. 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.

7.2.1 Transformation

Given an input protocol P, our transformation will compute a new protocol $\tilde{P}$ which consists of two phases. The transformation, using the informal Alice-Bob notation, is described in Figure 7.1. The k-tagging of u with tag, denoted $[u]_{\mathsf{tag}}$, consists in adding tag inside each encrypted subterm of u. Note that, the 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.

Let P be a k-party protocol.

$$P = \begin{cases} A_{i_1} \to A_{j_1} : & m_1 \\ \vdots & \\ A_{i_\ell} \to A_{j_\ell} : & m_\ell \end{cases} \qquad \tilde{P} = \begin{cases} \text{Phase 1} & & \text{Phase 2} \\ A_1 \to All : & \langle A_1, N_1 \rangle & A_{i_1} \to A_{j_1} : \quad [m_1]_\tau \\ \vdots & & \vdots \\ A_k \to All : & \langle A_k, N_k \rangle & A_{i_\ell} \to A_{j_\ell} : \quad [m_\ell]_\tau \\ \multicolumn{3}{l}{\text{where } \tau = \langle \mathsf{tag}_1, \ldots, \mathsf{tag}_k \rangle \text{ with } \mathsf{tag}_i = \langle A_i, N_i \rangle} \end{cases}$$

Figure 7.1 - Our transformation in Alice-Bob notation.

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: $\tau = \langle \langle A_1, N_1 \rangle, \ldots, \langle A_k, N_k \rangle \rangle$. Note that an active attacker may interfere with this initialization phase and may 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 prim-

itive. 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.

7.2.2 Main result

Our result states that if a protocol (obtained after application of our transformation) 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. The situation is however slightly more complicated than it may seem at first sight since there is an infinite number of honest sessions, which one would need to verify separately. Actually we can avoid this combinatorial explosion thanks to the following result [CLC04a]: when verifying secrecy properties it is sufficient to consider one single honest agent (which is allowed to "talk to herself"). Hence we can instantiate all the parameters with the same honest agent.

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. A more detailed discussion showing that *static tags* are not sufficient follows in Section 7.2.3. To obtain our composition result, we need also to forbid long-term secrets in plaintext position (even under an encryption).

Our result holds for the same class of protocols as the one presented in Section 7.1. We only prove this result for secrecy properties. However, since then, M. Arapinis has extended this result to deal with some other trace properties [Ara08]. The bound on the number of sessions that have to be considered depends on the underlying security property.

7.2.3 Other ways of tagging

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 (one obtained after applying our transformation) 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 [NS78], 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.

Finally, 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 those that 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]).

7.3 Composing protocols sharing passwords

While the absence of shared keys between different protocols is obviously desirable, it is not always possible or realistic. For example, *password-based protocols* are those in which a user picks a password which forms one of the secrets used in the protocol. It is unrealistic to assume that users never share the same passwords between different applications. We consider the situation in which secret data may be shared between protocols, and we particularly focus on password-based protocols. We investigate under what conditions we can guarantee that such protocols will not interfere with each other.

Additionally, the notion of security we consider is *resistance to guessing attacks* (see Section 4.3.1). If the attacker's interaction with the protocol during the first phase is limited to eavesdropping, then the attack is called *passive*; if the attacker can participate fully with the protocol, then it is *active*. In this section, we sum up the results fully developed in [25].

Related work. As we have seen in the two previous subsections, the problem of secure composition has been approached by several authors. However, none of the works cited above deals with composing resistance against guessing attacks. They consider secrecy in term of deducibility or authentication properties. To the best of our knowledge only S. Malladi *et al.* [MAFM02] have studied composition *w.r.t.* guessing attacks. They point out vulnerabilities that arise when the same password is used for different applications and develop a method to derive conditions that a protocol has to follow in order to be resistant against guessing attacks. However, applying their methods to particular protocols is not always straightforward. Moreover, their work relies on the definition of guessing attack due to Lowe [Low04] which relies on a particular set of cryptographic primitives. Our results are general and independent of the underlying equational theory.

7.3.1 Passive case

We first established 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 in accordance with the one stated in Definition 4.5. The second one is due to R. Corin *et al.* [CDE05]. The last definition is given in a composable way and establishes our composition result (see Corollary 7.1).

Proposition 7.1 *[25] Let ϕ be a frame such that $\phi \equiv \mathtt{new}\,\tilde{w}.\phi'$ and $\tilde{w}'$ be a sequence of fresh names. The three following statements are equivalent:*

1. $\mathtt{new}\,\tilde{w}.(\phi' \mid \{^{\tilde{w}}/_{\tilde{x}}\}) \sim \mathtt{new}\,\tilde{w}'.\mathtt{new}\,\tilde{w}.(\phi' \mid \{^{\tilde{w}'}/_{\tilde{x}}\})$,
2. $\phi' \sim \mathtt{new}\,\tilde{w}.\phi'$,
3. $\phi' \sim \phi'\{^{\tilde{w}'}/_{\tilde{w}}\}$.

Now, by relying on Proposition 7.1 (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 7.1 *[25] Let $\phi_i \equiv \mathtt{new}\,\tilde{w}.\phi_i'$ with $(i = 1, 2)$ be two frames such that $\mathtt{new}\,\tilde{w}.(\phi_1' \mid \phi_2')$ is also a frame (this can be achieved by using α-renaming).*

If ϕ_1 and ϕ_2 are resistant to guessing attacks against $\tilde{w}$ then $\mathtt{new}\ \tilde{w}.(\phi_1' \mid \phi_2')$ *is also resistant to guessing attacks against $\tilde{w}$.*

Applications. In the case of *password-only* protocols, *i.e.*, protocols that only share a password between different sessions and do not have any other common 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], which has also been analysed in [CDE05].

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

7.3.2 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. Hence, there is no hope to obtain a general composition result. To reach our goal, we consider a restricted class of protocols: the class of *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.*, h does not occur in any equation modelling 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.

Note that a protocol can be very easily transformed into a well-tagged protocol: if $\mathtt{new}\ w.A$ is a process resistant to guessing attacks against w, then the transformed process is obtained by replacing any occurrence of the password w by $\mathsf{h}(\alpha, w)$. We have shown that resistance against guessing attacks is preserved by our transformation. The simplicity of our transformation should also ensure that the functionalities of the protocol are preserved as well. A rigorous proof of this would require a formal definition of what it means to "preserve the functionalities" of a protocol.

Theorem 7.1 *[25] Let* $\mathtt{new}\ w.A'$ *be a process resistant to guessing attacks against w, then we have that* $\mathtt{new}\ w.(A'\{^{\mathsf{h}(\alpha,w)}/_{w}\})$ *is also resistant to guessing attacks against w.*

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. We consider an arbitrary equational theory E, provided there is no equation for h.

Theorem 7.2 *[25] 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* $\mathtt{new}\, w.(A_1 \mid A_2)$ *is a process (this can be achieved by using α-renaming).*

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

Assuming that any attack only uses a finite number of sessions, our composition result holds for an unbounded number of sessions, *i.e.* if two well-tagged protocols are separately resistant against guessing attacks for an unbounded number of sessions then their parallel composition is also resistant to guessing attacks for an unbounded number of sessions.

7.4 More related work and perspectives

As we have seen in Chapter 1, there are two large classes of models for studying the security of cryptographic protocols. On one hand, there are the symbolic models (also called Dolev-Yao model [DY81]) in which messages sent over the network are represented by terms and the attacker is modelled as a deduction system. This is the approach considered in this habilitation thesis. On the other hand, there are the computational models, in which the messages are bit-strings and the attacker is an arbitrary probabilistic polynomial time Turing machine. Our results clearly belong to the first approach. Moreover, our main goal was to provide syntactic criteria to ensure that different protocols (or different sessions coming from the same protocol) can be safely composed. However, composition has also been considered in the other approach yielding to an interesting paradigm, namely the *Universal Composability* framework [Can01], which we briefly discuss below.

First, it is important to notice that the results presented in this chapter do not allow one to compose protocols for the study of privacy-type properties. A natural extension consists in investigating whether parallel and sequential compositions preserve trace equivalence and more generally other behavioral equivalences that can be used to reason about security properties such as privacy. In particular, in Chapter 4, privacy in e-voting is stated by considering an arbitrary number of voters where two honest voters play a special role. It will be interesting to identify under which conditions we can safely restrict our analysis to a small number of voters. Note that in this situation, tagging as defined in Section 7.2 can not be applied.

Another line of work is represented by the Protocol Composition Logic (see *e.g.* [DMRSar]), which can be used to modularly prove security properties of protocols. In order to safely compose two protocols, one has to check that each protocol satisfies some invariant used in the security proof of the other protocol. While offering more flexibility, this criterion is not syntactic and needs to be checked each time by hand. Based on approximately ten years of experience with successive versions of the logic, PCL appears to scale well to industrial protocols. While the logic was originally developed for the symbolic model, a variant of the logic with similar reasoning principles has also been developed for the computational model.

Our work is also related to those of R. Canetti *et al.* who study universal composability of protocols [Can01] in a different context (cryptographic model). They consider composition in a broader sense than our composition result. Indeed, they both enable composition of a protocol with arbitrary other protocols and composition of a protocol with copies of itself. However, in the initial approach [Can01], they do not allow shared data between protocols, meaning that new encryption and signing keys have to be generated for each component. Composition theorems that allow joint states between protocols are proposed in further work (see *e.g.* [CR03, BCNP04, CDPW07]). Several limitations on the applicability of the results are listed in a recent paper of R. Küsters and M. Tuengerthal [KT08]. In [KT08], the authors propose a new construction for a joint state theorem in the context of asymmetric encryption

and signature only. This result allows in particular to compose a protocol with arbitrary protocols using the same kind of construction: they add an identifier in each encryption.

The universal composability paradigm has been quite successful in the computational approach. Hence, it seems natural to investigate the statement of general composition theorems (with joint states) in the symbolic settings.

Chapter 8

Conclusion and Perspectives

This habilitation thesis reports on a selection of my recent contributions to the verification of security protocols, from the development of algorithms to decide trace-based security properties in ad hoc routing protocols or security APIs (Part I), to the study of privacy-type security properties that play an important role in many modern applications such as electronic voting protocols (Part II). Moreover, because of the complexity of the verification problem and the variety of the envisioned applications, it becomes important to consider modularity issues. Some results on this aspect have been presented at the end of this thesis (Part III).

At the end of each chapter, some further developments for the given chapter have already been proposed. We recall the most important ones, and give also further perspectives. They are mostly oriented to deal with new emerging applications such as electronic voting and applications involving mobile devices, *e.g.* RFID tags, routing protocols in mobile ad hoc networks, and safety-related applications in vehicular ad hoc networks.

In Section 8.1, several features needed to model up-to-date applications such as the one mentioned above are presented. This raises several challenges since the existing verification techniques have to be extended accordingly. As location tracking capabilities of mobile devices are increasing, problems related to user privacy arise. Indeed user's position and preferences constitute personal information and improper use of them violates user's privacy. This raises several issues that are developed in Section 8.2. Lastly, to ensure security even when several applications are running concurrently and to analyse protocols that become more and more complex, composition is an important aspect that deserves to be further studied (see Section 8.3).

8.1 Security issues in mobile ad hoc network

Today, many applications rely on mobile devices such as smart phones, RFID tags, ... For instance, several countries have equipped their motorways with RFID payment systems (*e.g.* Telepass in Italy, AutoPASS in Norway), or their public transport with RFID passes such as Navigo in France. However, the use of RFID technology raises important security issues and has engendered considerable controversy. Indeed, tags, which are world-readable, pose a risk to personal location privacy. Many privacy organizations have expressed concerns in the context of ongoing efforts to embed RFID tags in consumer products. It is thus important to develop methods and verification tools that are suitable to analyse these applications.

8.1.1 Modelling issues

Mobile ad hoc network applications have some specificities that prevent them to be analysed by existing automated verification tools designed for analysing key-exchange and authentication protocols (*e.g.* AVISPA [ABB+05], ProVerif [Bla01], ...). To develop verification algorithms that are suitable for mobile ad hoc network applications, it is important to better understand what are the features that play an important role and that need to be modelled in a more accurate way. As usual, we have to achieve two antagonist goals. On the one hand, models have to be as fine grained and expressive as possible in order to better reflect protocol behaviours. One the other hand, models have to remain relatively simple in order to allow the design of automatic decision procedures.

Communications. In mobile ad hoc network, there is no underlying infrastructure and communications are done wireless. Moreover, the main mode of communication is broadcasting and reception of a broadcast is restricted to single-hop neighbours. In this setting, it is thus physically impossible to prevent a message issued by a node A to reach a node B that is in the range of A.

Attacker. Until now, the classical Dolev-Yao attacker model [DY81] has been successfully used to analyse security protocols. However, in the context of mobile ad hoc network applications, considering an attacker who controls the entire communication network leads to the discovery of a number of unrealistic attacks. Hence, we have to consider a distributed attacker who controls some nodes of the network. This results in a distributed attacker with communication abilities that are restricted, but more realistic than the classical Dolev-Yao attacker.

Security properties. As already mentioned, applications involving mobile devices raise some important issues about privacy. This aspect is developed in Section 8.2. Some other attacks need to be further studied, *e.g.* Sybil attacks [Dou02], rushing attacks [HPJ03], ... This will conduct us to adapt our models. For instance, rushing attacks will require a precise modelling of the time of transmission of the messages.

Tests on incoming messages. In mobile ad hoc network, a fundamental building block is neighbourhood discovery protocols [PPS+08]. A node must be able to determine or verify its direct communication partners within a communication network. To model in an abstract way the checks performed by the nodes involved in these protocols, we need to consider new constructions such as those described in Section 2.3. Moreover, in some routing protocols (*e.g.* endairA [BV04], Ariadne [HPJ05]), during the reply phase, the nodes perform a check on their incoming message. Typically, the reply will contain a list (the route) and a message built recursively on this list. As a consequence, the test performed by a node can not be modelled in an accurate way using pattern-matching only since the size of the list is not known *a priori*. Other examples of protocols performing recursive operations are certification paths for public keys (see *e.g.* X.509 certification paths [HFP98]) or right delegation in distributed systems [Aur99]. Again, to model protocols manipulating lists and/or relying on recursive tests, we have to extend existing models with new constructions.

Mobility. Taking into account mobility is crucial for some applications. Because of the difficulties of the underlying verification problem, very few works take this aspect into account. For instance, in [NH06], the topology of the network is represented by a set of graphs and it is assumed that the current topology could be any graph in this set. Hence, mobility is modelled in a very restrictive way. Despite the challenges raised by mobility from the verification point of view, mobility raises also some important issues from the modelling point of view. Indeed, what does it mean for a routing protocol to be secure if the underlying topology is changing? For instance, we can not expect that the route received by the source is the right one if the topology has changed after the nodes have sent their replies.

A challenge would be to establish a model in the style of the applied pi calculus allowing us to take into account a variety of cryptographic primitives and the different aspects mentioned above. As a first step towards this goal, it would be interesting to develop some models dedicated to some applications. This will help us to precisely identify all the crucial aspects that are behind these new emerging applications.

8.1.2 Verification issues

In order to be able to analyse mobile ad hoc network applications, we have to adapt the existing verification algorithms and/or to develop new ones from scratch. Indeed, the existing techniques do not allow one for instance to reason about mobility or neighbourhood checks. Before developing practical algorithms, it is important to identify the frontier between decidability and undecidability. In particular, among the different aspects mentioned above, we have to identify those that will lead to an undecidable model. If possible, it would be interesting to identify reasonable conditions under which the verification problem would become decidable.

Bounded number of sessions. In the context of a bounded number of sessions, many decidability and complexity results have been obtained for classical protocols (see *e.g.* [RT03, MS05]). In particular, constraint solving techniques have been quite successful (see *e.g.* [MS03, CLCZ10]) and have lead to practical algorithms that are now implemented, *e.g.* OFMC [BMV05]. This method has appeared to be flexible enough to integrate some of the aspects mentioned above, *e.g.* neighbourhood checks, and some simple constructions on lists. This line of research deserves further investigations. For the moment, the decision procedures we obtained are highly non-deterministic and can not lead to any practical algorithm. Thus, it would be interesting to see how to obtain an implementable procedure that could be integrated in the existing tools.

Unbounded number of sessions. Even if it is well-known that the verification problem is undecidable in this setting for classical protocols, the modelling through Horn clauses has been successful and has lead to some automated tool (*e.g.* ProVerif). It seems relevant to investigate this approach in the context of mobile ad hoc networks. This will require over-approximating the protocols and probably to develop new resolution techniques or at least to adapt the existing ones. This will allow us to obtain verification procedures in the spirit of the one implemented in the ProVerif tool.

Reduction results. As already discussed, the topology of the network plays an important role, especially in the routing application. Instead of developing results to decide the existence of a topology that is vulnerable to an attack, we could try to obtain first some reduction results. This will allow us to simplify our analysis by discarding some of the topologies, concentrating on those that could lead to an attack. Moreover, in this new setting, the attacker model we have to consider is quite complex. Clearly, in the context of routing protocols, considering two attackers (each of them controlling one node) is stronger than considering only one attacker. However, is it really useful to consider more than two attackers?

8.2 Privacy-type security properties

We began the study of this topic few years ago through the electronic voting application. However, privacy is a general requirement that needs to be studied in different contexts. For instance, a person who carries an RFID tag (in a cloth or simply because she carries her electronic passport) can be tracked. This raises a privacy issue. The term privacy is a generic word to represent several concepts that are formally modelled in different ways. Nevertheless, they are often modelled by relying on the notion of observational equivalence. Indeed, in general, we need to formalize the fact that the attacker can not see the difference between two slightly different situations. For instance, in the context of RFID protocols, an important privacy issue is to ensure untraceability, meaning roughly that an attacker can not distinguish a scenario where a same tag is involved in several sessions from one that involved different tags.

8.2.1 More algorithms for analysing equivalence-based properties

Even in the context of electronic voting, the results obtained so far are not completely satisfactory. Indeed, among the most classical electronic voting protocols (*e.g.* [FOO92a, Oka96]), some of them can still not be analysed by existing tools. Many results already exist to study static equivalence (see Chapter 5). However, it is important to develop further algorithms to deal with cryptographic primitives such as trapdoor bit commitment and re-encryption, and then to develop verification tools allowing us to analyse privacy-type properties in this context.

Unbounded number of sessions. For an unbounded number of sessions, the problem of deciding observational equivalence is well-known to be undecidable. Nevertheless, we have seen that the ProVerif tool tackles this problem by providing a semi-decision procedure that provides some useful results in some cases. To achieve this, ProVerif relies on a strong notion of equivalence. This notion is however too strong for the study of many privacy-type properties, and we thus need to relax this notion. Another limitation of this tool is the fact that it is not able to deal with primitives such as re-encryption or trapdoor bit commitment schemes. It seems however possible to improve the ProVerif tool such that it will handle more primitives [KT11] and/or it will conclude in more cases by refining the notion of equivalence [Smy10]. These directions need to be further investigated.

Bounded number of sessions. Another promising approach is to reuse the constraint solving approach that has been quite successful to analyse trace-based security properties.

This is one of the approaches developed in Chapter 6. However, it remains to lift the decision procedure from constraint systems to processes. A naive approach as the one described in [21] is not sufficient. First, because the resulting algorithm will not be implementable. Second, because this approach works only for a certain class of processes, namely the class of *simple processes*. Our goal is to go beyond this class and to propose an algorithm that is efficient in practice. This will allows us to deal with some protocols that are out of reach of the existing tools and even out of scope of the exiting decision procedures, *e.g.* private authentication protocol [AF04], electronic passport protocol [ACRR10].

8.2.2 More cryptographic primitives

In the context of equivalence-based properties, the existing algorithms do not allow us to go beyond the subterm convergent equational theories. However, as we have already discussed, electronic voting protocols often rely on more complex primitives in order to achieve their goals [FOO92a, Oka96], and RFID protocols that have power-consumption constraints often used the exclusive-or operator as a cryptographic primitive [vDR09]. All these primitives are out of scope of the existing algorithms.

Convergent equational theories. Several procedures already exist to deal with subterm convergent equational theories (see *e.g.* [Bau05, CR11]). The ProVerif tool also works quite well in this setting. However, for instance, the homomorphic encryption theory, for which static equivalence is decidable and efficient tools already exist, has not been studied in the active setting.

Monoidal equational theories. RFID protocols often rely on the exclusive-or operator (see *e.g.* [vDR09]). It is thus important to extend the existing decision procedures for static equivalence to deal with monoidal equational theories (including the exclusive-or theory) in the active setting. A first step could be to study the notion of equivalence introduced by M. Baudet [Bau05]. This notion is simpler to reason with since it only requires to study symbolic equivalence of two constraint systems that only differ at one place. This first step will allow us to derive interesting procedures to decide guessing attacks.

Combination. Regarding static equivalence, results allowing us to combine non-disjoint theories are still missing. To the best of my knowledge, no combination result exist to deal with equivalence-based properties in the active setting. A first step could be to develop a combination algorithm for the problem studied in [Bau05] in order to decide resistance against guessing attacks for more complex equational theories. A combination algorithm for the problem of observational equivalence will allow us to obtain decision procedures to analyse protocols such as RFID protocols that often rely on exclusive or and hash functions.

8.3 Composition and refinement

As we have seen, there exist several formal methods to establish that a security protocol achieves some security goals. However, their applicability is limited to relatively small protocols that run in isolation. Actually most of the existing protocols are not meant to be executed in isolation (*e.g.* the goal of a key-exchange protocol is to establish a key that will

be used by some other protocols) and they are often composed of several sub-protocols. Thus, the current practice is that large or composed protocols are not formally verified. We would like to remedy this situation, and make formal methods applicable to real life systems.

Another reason for studying this problem of security-proof modularity is to better understand the works that have been done in the area of computational security. Indeed, in this framework, a huge amount of work has been devoted to universal composability (see *e.g.* [Can01]).

Several composition results have been stated in Chapter 7. More precisely, for a set of names $\tilde{n}$ (typically the public and private keys or some passwords), a class of protocols, an attacker model, and a class of security properties, we give some conditions under which the security of two individual protocols (or two sessions coming from the same protocol) $\mathtt{new}\,\tilde{n}.P_1$ and $\mathtt{new}\,\tilde{n}.P_2$ implies the security of the combined protocol $\mathtt{new}\,\tilde{n}.(P_1 \mid P_2)$.

8.3.1 Composition

The results stated in Chapter 7 only concern trace-based security properties. Moreover, those presented in Section 7.1 and Section 7.2 assume a fixed set of cryptographic primitives. Lastly, we consider two different kinds of composition that have some common features, namely protocols composition and sessions composition. Note also that in all the three composition results that we have established, we achieve our goal by tagging the messages. Even if the way we tagged the messages are different depending on the composition result we want to achieve, they have some common features. It would be interesting to better understand the properties that are required on the message that is used to tag the protocol messages, and to obtain more general composition results.

For instance, resistance against guessing attacks is not the only purpose of a password-based protocol. Thus, it would be interesting to ensure also that secrecy and authentication properties are preserved when we compose password-based protocols.

Concerning privacy-type properties, it is well-known that composition works when the processes do not rely on some shared secrets. However, there is no result allowing us to derive

$$\mathtt{new}\,\tilde{n}.(P_1 \mid P_2) \approx \mathtt{new}\,\tilde{n}.(Q_1 \mid Q_2)$$

from the equivalences $\mathtt{new}\,\tilde{n}.P_1 \approx \mathtt{new}\,\tilde{n}.Q_1$ and $\mathtt{new}\,\tilde{n}.P_2 \approx \mathtt{new}\,\tilde{n}.Q_2$. This kind of composition results will be very useful. For instance, this could allow us to establish privacy in presence of two honest voters or untraceability in presence of two different tags, and to obtain guarantee in a setting that involves an arbitrary number of voters or tags.

Lastly, in some context (*e.g.* routing protocols or group protocols), the transformations described in Chapter 7 can not be applied since the number of participants is not known *a priori*. Nevertheless, we may want to avoid interaction between different sessions and to restrict our attention to the study of a single session. The study of this aspect would allow us to provide some prudent engineering principles in the style of [AN96] that are applicable in the more general setting of group protocols or routing protocols.

8.3.2 Refinement

Security protocols used in practice are more and more complex and it is difficult to analyse them entirely. Thus, some parts are usually abstracted away. For instance, we may assume that a symmetric key has been magically established between different parties, or we may

assume that a communication has been performed on a private or authenticated channel. However, the interactions between the main protocol and the key-establishment protocol can be damaging for the security of the resulting protocol.

In order to obtain guarantees on the whole protocol without analysing it entirely, we have to develop refinement results. These results will give us conditions under which a part of a protocol can be safely abstracted away (and in which way) without missing any attack. Recently, several attempts have been done to obtain results in this direction (see *e.g.* [ACS+08, CC10]. However, these results do not allow one for instance to compose protocols that rely on some primitives such as the exclusive-or operator and only consider some particular trace-based security properties. It would be interesting to study refinement in a more general setting. In particular, privacy in electronic voting is a quite complex security property to establish and we often rely on the existence of private channels to model this kind of protocols. In order to justify this abstraction, refinement results for equivalence-based properties are needed.

8.3.3 Symbolic UC

The universal composability paradigm has been quite successful in the computational approach allowing to model both composition and refinement (see *e.g.* [Can01, BCNP04, KT08]) However, both the composition theorems and the proofs that security primitives satisfy the hypotheses of these theorems are quite delicate in the computational world.

The idea of UC is not, however, restricted to the computational setting; instead, one can see it as a refinement relation on protocols and programs that preserves security and is composable. Moreover, in order to get computational security guarantees, we have seen that there is an alternative method: using soundness theorems in the style of [AR00]. Hence, assuming appropriate soundness results, the modularity of symbolic security proofs would have therefore an important impact in the computational security community. Therefore, it seems natural to investigate the statement of general composition theorems in the UC style but considering the symbolic setting instead of the compuational one.

Actually, we have already obtained a first result in this direction [19]. While the resulting theorems appear to be the natural counterpart of their computational versions, this work brings the benefits of the secure composition theorems associated with simulation-based security into the symbolic world, and opens the path to the analysis of more sophisticated protocols that can naturally be specified by the behavior of an ideal functionality.

My publications

A complete and up to date version of my list of publications is available on line.

`http://www.lsv.ens-cachan.fr/~delaune/mes-publis.php`

Chapters in books

[1] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols: A taster. In D. Chaum, M. Jakobsson, R. L. Rivest, P. Y. A. Ryan, J. Benaloh, M. Kutyłowski, and B. Adida, editors, *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer, May 2010.

Journals

[2] V. Cortier and S. Delaune. Decidability and combination results for two notions of knowledge in security protocols. *Journal of Automated Reasoning*, 2011. To appear.

[3] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 2011. To appear.

[4] S. Delaune, S. Kremer, and G. Steel. Formal analysis of PKCS#11 and proprietary extensions. *Journal of Computer Security*, 18(6):1211–1245, Nov. 2010.

[5] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 18(2):317–377, Mar. 2010.

[6] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[7] V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1):1–36, Feb. 2009.

[8] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206(2-4):312–351, Feb.-Apr. 2008.

[9] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[10] S. Delaune. An undecidability result for AGh. *Theoretical Computer Science*, 368(1-2):161–167, Dec. 2006.

[11] S. Delaune and F. Jacquemard. Decision procedures for the security of protocols with probabilistic encryption against offline dictionary attacks. *Journal of Automated Reasoning*, 36(1-2):85–124, Jan. 2006.

[12] S. Delaune. Easy intruder deduction problems with homomorphisms. *Information Processing Letters*, 97(6):213–218, Mar. 2006.

Conferences

[13] M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for anonymous location based services. In *Proceedings of the Workshop on Theory of Security and Applications (TOSCA'11)*, Saarbrücken, Germany, Mar.-Apr. 2011. To appear.

[14] S. Delaune, S. Kremer, M. D. Ryan, and G. Steel. A formal analysis of authentication in the TPM. In P. Degano, S. Etalle, and J. Guttman, editors, *Revised Selected Papers of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, volume 6561 of *Lecture Notes in Computer Science*, pages 111–125, Pisa, Italy, Sept. 2010. Springer.

[15] M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In D. Gritzalis and B. Preneel, editors, *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *Lecture Notes in Computer Science*, pages 55–70, Athens, Greece, Sept. 2010. Springer.

[16] M. Arnaud, V. Cortier, and S. Delaune. Modeling and verifying ad hoc routing protocols. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 59–74, Edinburgh, Scotland, UK, July 2010. IEEE Computer Society Press.

[17] V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In J. Giesl and R. Haehnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426, Edinburgh, Scotland, UK, July 2010. Springer-Verlag.

[18] S. Bursuc, S. Delaune, and H. Comon-Lundh. Deducibility constraints. In A. Datta, editor, *Proceedings of the 13th Asian Computing Science Conference (ASIAN'09)*, volume 5913 of *Lecture Notes in Computer Science*, pages 24–38, Seoul, Korea, Dec. 2009. Springer.

[19] S. Delaune, S. Kremer, and O. Pereira. Simulation based security in the applied pi calculus. In R. 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, Dec. 2009. Leibniz-Zentrum für Informatik.

[20] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In R. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*, Lecture Notes in Artificial Intelligence, pages 355–370, Montreal, Canada, Aug. 2009. Springer.

[21] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276, Port Jefferson, NY, USA, July 2009. IEEE Computer Society Press.

[22] M. Baudet, V. Cortier, and S. Delaune. YAPA: A generic tool for computing intruder knowledge. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163, Brasília, Brazil, June-July 2009. Springer.

[23] R. Chadha, S. Delaune, and S. Kremer. Epistemic logic for the applied pi calculus. In D. Lee, A. Lopes, and A. 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.

[24] M. Arapinis, S. Delaune, and S. Kremer. From one session to many: Dynamic tags for security protocols. In I. Cervesato, H. Veith, and A. 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, Nov. 2008. Springer.

[25] S. Delaune, S. Kremer, and M. 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.

[26] S. Delaune, S. Kremer, and G. 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.

[27] S. Delaune, M. D. Ryan, and B. Smyth. Automatic verification of privacy properties in the applied pi-calculus. In Y. Karabulut, J. Mitchell, P. Herrmann, and C. D. Jensen, editors, *Proceedings of the 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security (IFIPTM'08)*, volume 263 of *IFIP Conference Proceedings*, pages 263–278, Trondheim, Norway, June 2008. Springer.

[28] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi-calculus. In V. Arvind and S. 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, Dec. 2007. Springer.

[29] V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In V. Arvind and S. 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 352–363, New Delhi, India, Dec. 2007. Springer.

[30] V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In N. Dershowitz and A. Voronkov, editors, *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 196–210, Yerevan, Armenia, Oct. 2007. Springer.

[31] S. Delaune, H. Lin, and Ch. Lynch. Protocol verification via rigid/flexible resolution. In N. Dershowitz and A. Voronkov, editors, *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 242–256, Yerevan, Armenia, Oct. 2007. Springer.

[32] M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In F. Wolter, editor, *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117, Liverpool, UK, Sept. 2007. Springer.

[33] V. Cortier, S. Delaune, and G. Steel. A formal theory of key conjuring. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 79–93, Venice, Italy, July 2007. IEEE Computer Society Press.

[34] S. Bursuc, H. Comon-Lundh, and S. Delaune. Deducibility constraints, equational theory and electronic money. In H. Comon-Lundh, C. Kirchner, and H. Kirchner, editors, *Rewriting, Computation and Proof — Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of his 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, pages 196–212, Cachan, France, June 2007. Springer.

[35] S. Bursuc, H. Comon-Lundh, and S. Delaune. Associative-commutative deducibility constraints. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 of *Lecture Notes in Computer Science*, pages 634–645, Aachen, Germany, Feb. 2007. Springer.

[36] S. Delaune, S. Kremer, and M. 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.

[37] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis in presence of a homomorphism operator and *exclusive or*. In M. Buglesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06) — Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 132–143, Venice, Italy, July 2006. Springer.

[38] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307, Nara, Japan, Apr. 2005. Springer.

[39] S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In V. Atluri, B. Pfitzmann, and P. McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, Washington, D.C., USA, Oct. 2004. ACM Press.

[40] S. Delaune and F. Jacquemard. A theory of dictionary attacks and its complexity. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 2–15, Asilomar, Pacific Grove, California, USA, June 2004. IEEE Computer Society Press.

Theses

[41] S. Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, June 2006.

[42] S. Delaune. Vérification de protocoles de sécurité dans un modèle de l'intrus étendu. Rapport de DEA, DEA Programmation, Paris, France, Sept. 2003.

Contract reports

[43] S. Delaune. Algorithms for observational equivalence. Deliverable AVOTE 2.2, (ANR-07-SESU-002), Jan. 2011. 118 pages.

[44] S. Delaune and S. Kremer. Spécificités des protocoles de vote électronique. Deliverable AVOTE 1.1 (ANR-07-SESU-002), Jan. 2009. 8 pages.

[45] S. Delaune and F. Klay. Synthèse des expérimentations. Technical Report 10, projet RNTL PROUVÉ, May 2007. 10 pages.

[46] F. Klay, L. Bozga, Y. Lakhnech, L. Mazaré, S. Delaune, and S. Kremer. Retour d'expérience sur la validation du vote électronique. Technical Report 9, projet RNTL PROUVÉ, Nov. 2006. 47 pages.

[47] S. Delaune, F. Klay, and S. Kremer. Spécification du protocole de vote électronique. Technical Report 6, projet RNTL PROUVÉ, Nov. 2005. 19 pages.

[48] L. Bozga, S. Delaune, F. Klay, and L. Vigneron. Retour d'expérience sur la validation du porte-monnaie électronique. Technical Report 5, projet RNTL PROUVÉ, Mar. 2005. 29 pages.

[49] V. Bernat, H. Comon-Lundh, V. Cortier, S. Delaune, F. Jacquemard, P. Lafourcade, Y. Lakhnech, and L. Mazaré. Sufficient conditions on properties for an automated verification: theoretical report on the verification of protocols for an extended model of the intruder. Technical Report 4, projet RNTL PROUVÉ, Dec. 2004. 33 pages.

[50] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. Technical Report 2, projet RNTL PROUVÉ, June 2004. 19 pages.

[51] L. Bozga, S. Delaune, F. Klay, and R. Treinen. Spécification du protocole de porte-monnaie électronique. Technical Report 1, projet RNTL PROUVÉ, June 2004. 12 pages.

Other publications

[52] M. Dahl, S. Delaune, and G. Steel. Formal analysis of privacy for vehicular mix-zones. In V. Cortier, M. D. Ryan, and V. Shmatikov, editors, *Proceedings of the Workshop on Foundations of Security and Privacy (FCS-PrivMod'10)*, Edinburgh, Scotland, UK, July 2010.

[53] S. Bursuc, H. Comon-Lundh, and S. Delaune. Deducibility constraints and blind signatures. Research Report LSV-10-24, Laboratoire Spécification et Vérification, ENS Cachan, France, Nov. 2010. 32 pages.

[54] S. Delaune, S. Kremer, M. D. Ryan, and G. Steel. A formal analysis of authentication in the TPM (short paper). In V. Cortier and K. Chatzikokolakis, editors, *Preliminary Proceedings of the 8th International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'10)*, Paris, France, Aug. 2010.

[55] V. Cheval, H. Comon-Lundh, and S. Delaune. A decision procedure for proving observational equivalence. In M. Boreale and S. Kremer, editors, *Preliminary Proceedings of the 7th International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'09)*, Bologna, Italy, Oct. 2009.

[56] M. Arnaud, V. Cortier, and S. Delaune. Modeling and verifying ad hoc routing protocol. In H. Comon-Lundh and C. Meadows, editors, *Preliminary Proceedings of the 4th International Workshop on Security and Rewriting Techniques (SecReT'09)*, pages 33–46, Port Jefferson, NY, USA, July 2009.

[57] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In H. Comon-Lundh and C. 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.

[58] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. In D. Goria and C. Palamidessi, editors, *Preliminary Proceedings of the 5th International Workshop on Security Issues in Concurrency (SecCo'07)*, Lisbon, Portugal, Sept. 2007.

[59] V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In P. Degano, R. Küsters, L. Viganò, and S. Zdancewic, editors, *Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07)*, pages 63–80, Wrocław, Poland, July 2007.

[60] S. Delaune, H. Lin, and Ch. Lynch. Protocol verification via rigid/flexible resolution. In S. Ghilardi, U. Sattler, V. Sofronie-Stokkermans, and A. Tiwari, editors, *Proceedings of the Workshop on Automated Deduction: Decidability, Complexity, Tractability (ADDCT'07)*, pages 2–16, Bremen, Germany, July 2007.

[61] S. Delaune, S. Kremer, and M. 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.

[62] S. Delaune, S. Kremer, and M. D. Ryan. Receipt-freeness: Formal definition and fault attacks (extended abstract). In *Proceedings of the Workshop Frontiers in Electronic Elections (FEE 2005)*, Milan, Italy, Sept. 2005.

[63] S. Delaune and F. Jacquemard. Narrowing-based constraint solving for the verification of security protocols. In M. Kohlhase, editor, *Proceedings of the 18th International Workshop on Unification (UNIF'04)*, Cork, Ireland, July 2004.

[64] S. Delaune. Intruder deduction problem in presence of guessing attacks. In M. Rusinowitch, editor, *Proceedings of the Workshop on Security Protocols Verification (SPV'03)*, pages 26–30, Marseilles, France, Sept. 2003.

Bibliography

[AB03] M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. In *Proceedings of the 10th International Symposium (SAS'03)*, volume 2694 of *Lecture Notes in Computer Science*, pages 316–335. Springer, 2003.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The Avispa tool for the automated validation of internet security protocols and ap plications. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *Lecture Notes in Computer Science*, 2005.

[AC02] R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR'02)*, volume 2421 of *Lecture Notes in Computer Science*, pages 499–514, Brno (Czech Republic), 2002. Springer-Verlag.

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

[ACRR10] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.

[ACS+08] S. Andova, C. Cremers, K. G. Steen, S. Mauw, S. M. lsnes, and S. Radomirović. Sufficient conditions for composing security protocols. *Information and Computation*, 206(2-4):425–459, 2008.

[AD07] M. Arapinis and M. 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.

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, London (United Kingdom), 2001. ACM Press.

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

[AG97] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 36–47, Zurich (Switzerland), 1997. ACM Press.

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[ALV02] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[AMRV06] P. Adão, P. Mateus, T. Reis, and L. Viganò. Towards a quantitative analysis of security protocols. *Electronic Notes in Theoretical Computer Sciences*, 164(3):3–25, 2006.

[AN96] M. Abadi and R. M. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography: the computational soundness of formal encryption. In *Proceedings of the 1rst IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, 2000.

[Ara08] M. Arapinis. *Sécurité des protocoles cryptographiques : décidabilité et résultats de réduction*. Thèse de doctorat, Université Paris 12, Créteil, France, November 2008.

[Aur99] T. Aura. Distributed access-rights management with delegation certificates. In *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 211–235. 1999.

[Baa89] F. Baader. Unification in commutative theories. *Journal of Symbolic Computation*, 8(5):479–497, 1989.

[Baa93] F. Baader. Unification in commutative theories, Hilbert's basis theorem, and Gröbner bases. *Journal of the ACM*, 40(3):477–503, 1993.

[BAF08] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[Bau05] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[Bau07] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Phd thesis, École Normale Supérieure de Cachan, France, 2007.

[Bau08] M. Baudet. YAPA (Yet Another Protocol Analyzer), 2008. `http://www.lsv.ens-cachan.fr/~baudet/yapa/`.

[BCC04] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pages 132–145, New York, USA, 2004. ACM Press.

[BCdH10] M. Bruso, K. Chatzikokolakis, and J. den Hartog. Formal verification of privacy for RFID systems. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*. IEEE Computer Society Press, 2010.

[BCFS10] M. Bortolozzo, M. Centenaro, R. Focardi, and G. Steel. Attacking and fixing PKCS#11 security tokens. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 260–269, Chicago, Illinois, USA, 2010. ACM Press.

[BCNP04] B. Barak, R. Canetti, J. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS'04)*, pages 186–195. IEEE Computer Society Press, 2004.

[BG01] D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. *Journal of the ACM*, 48(1):70–109, 2001.

[BHM08] M. Backes, C. Hritcu, and M. Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium, (CSF'08)*, pages 195–209. IEEE Computer Society Press, 2008.

[BJPV09] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*, pages 39–48. IEEE Computer Society Press, 2009.

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96, Cape Breton (Canada), 2001. IEEE Computer Society Press.

[Bla02] B. Blanchet. From secrecy to authenticity in security protocols. In *Proceedings of the 9th International Symposium, (SAS'02)*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359, Madrid (Spain), 2002. Springer.

[Bla04] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Proceedings of the 25th Symposium on Security and Privacy (S&P'04)*, Berkeley, CA (USA), 2004. IEEE Computer Society Press.

[BM92] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the Symposium on Security and Privacy (S&P'92)*, pages 72–84. IEEE Computer Society Press, 1992.

[BMV05] D. A. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, 2005.

[BMV10] D. Benetti, M. Merro, and L. Viganò. Model checking ad hoc network routing protocols: ARAN vs. endairA. In *Proceedings of the 8th IEEE International Conference on Software Engineering and Formal Methods (SEFM'10)*. IEEE Computer Society Press, 2010.

[Bon01] M. Bond. Attacks on cryptoprocessor transaction sets. In *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*, volume 2162 of *Lecture Notes in Computer Science*, pages 220–234, Paris (France), 2001. Springer-Verlag.

[Bor01] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681, Crete (Greece), 2001. Springer-Verlag.

[BP03] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *Lecture Notes in Computer Science*, pages 136–152. Springer, 2003.

[BRC09] M. Berrima, N. B. Rajeb, and V. Cortier. Deciding knowledge in security protocols under some e-voting theories. Technical Report RR-6882, INRIA, April 2009.

[BS96] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computation*, 21(2):211–243, 1996.

[BS98] F. Baader and K. U. Schulz. Combination of constraint solvers for free and quasi-free structures. *Theoretical Computer Science*, 192(1):107–161, 1998.

[BS03] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the 26th Symposium on Theory of Computing (STOC'94)*, pages 544–553. ACM Press, 1994.

[BV04] L. Buttyán and I. Vajda. Towards Provable Security for Ad Hoc Routing Protocols. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04)*, pages 94–105, New York, NY, USA, 2004. ACM.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 136–145. IEEE Computer Society Press, 2001.

[CB03] R. Clayton and M. Bond. Experience using a low-cost FPGA design to crack DES keys. In *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded System (CHES'02)*, volume 2523 of *Lecture Notes in Computer Science*, pages 579–592, Redwood Shores (CA, USA), 2003. Springer.

[CBC10] B. Conchinha, D. Basin, and C. Caleiro. Efficient decision procedures for message deducibility and static equivalence. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, volume 6561 of *Lecture Notes in Computer Science*, pages 39–49, Pisa, Italy, 2010. Springer.

[CC10] Ş. Ciobâcă and V. Cortier. Protocol composition for arbitrary primitives. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 322–336, Edinburgh, Scotland, UK, July 2010. IEEE Computer Society Press.

[CCA06] *CCA Basic Services Reference and Guide*, October 2006. Available online `http://www-03.ibm.com/security/cryptocards/pdfs/bs327.pdf`.

[CCG+02] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of the International Conference on Computer-Aided Verification (CAV'02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364, Copenhagen, Denmark, July 2002. Springer.

[CDE05] R. Corin, J. Doumen, and S. Etalle. Analysing password protocol security against off-line dictionary attacks. *Electonic Notes in Theoretical Computer Science*, 121:47–63, 2005.

[CDL+99] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proceedings of the 12th Computer Security Foundations Workshop (CSFW'99)*, pages 55–69, Mordano (Italy), 1999. IEEE Computer Society Press.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *Proceedings of the 4th Theory of Cryptography Conference, (TCC'07)*, Lecture Notes in Computer Science, pages 61–85. Springer, 2007.

[CGL+10] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen. Principles of remote attestation. *International Journal of Information Security*, To Appear, 2010.

[Cio09] Ş. 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`. 1997.

[CJM03] E. M. Clarke, S. Jha, and W. R. Marrero. Efficient verification of security protocols using partial-order reductions. *International Journal on Software Tools for Technology Transfer*, 4(2):173–188, 2003.

[CJS+08] I. Cervesato, A. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key kerberos. *Information and Computation*, 206(2-4):402–424, 2008.

[CKRT03a] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135, Mumbai (India), 2003. Springer-Verlag.

[CKRT03b] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 261–270, Ottawa (Canada), 2003. IEEE Computer Society Press.

[CKS07] V. Cortier, G. Keighren, and G. Steel. Automatic analysis of the security of xor-based key management schemes. In *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, Lecture Notes in Computer Science, pages 538–552, Braga (Portugal), 2007. Springer-Verlag.

[CKW10] V. Cortier, S. Kremer, and B. Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *Journal of Automated Reasoning*, 2010. To appear.

[CLC03] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 148–164, Valencia (Spain), 2003. Springer-Verlag.

[CLC04a] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1-3):51–71, 2004.

[CLC04b] H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Theoretical Computer Science, to appear*, 2004.

[CLCZ10] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties of cryptographic protocols. application to key cycles. *Transaction on Computational Logic*, 11(2), 2010.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 271–280, Ottawa (Canada), 2003. IEEE Computer Society Press.

[Clu03a] J. Clulow. On the security of PKCS#11. In *Proceedings of the 5th International Worshop on Cryptographic Hardware and Embedded Systems (CHES'03)*, volume 2779 of *Lecture Notes in Computer Science*, pages 411–425, Cologne, Germany, 2003. Springer.

[Clu03b] J. Clulow. The design and analysis of cryptographic APIs for security devices. Master's thesis, University of Natal, Durban, South Africa, 2003. Chapter 3.

[CM06] J. Courant and J.-F. Monin. Defending the bank with a proof assistant. In *Proceedings of Workshop on Issues in the Theory of Security (WITS '06)*, Vienna (Austria), March 2006.

[Cor06] R. Corin. *Analysis models for security protocols*. Phd thesis, University of Twente, 2006.

[CR03] R. Canetti and T. Rabin. Universal composition with joint state. In *Proceedings of the 23rd International Cryptology Conference (CRYPTO'03)*, Lecture Notes in Computer Science, pages 265–281. Springer, 2003.

[CR05] Y. Chevalier and M. Rusinowitch. Combining intruder theories. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651, Lisboa (Portugal), 2005. Springer.

[CR06] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In *Proceedings of the 17th International Conference on Term Rewriting and Applications (RTA'06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122, Seattle, USA, 2006. Springer.

[CR08a] L. Chen and M. D. Ryan. Offline dictionary attack on TCG TPM weak authorisation data, and solution. In *Future of Trust in Computing*, 2008.

[CR08b] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. *Information and Computation*, 206(2-4):352–377, 2008.

[CR09] L. Chen and M. Ryan. Attack, solution and verification for shared authorisation data in TCG TPM. In *Proceedings of the 6th International Workshop on Formal Aspects in Security and Trust (FAST'09)*, pages 201–216, 2009.

[CR11] Y. Chevalier and M. Rusinowitch. Decidability of symbolic equivalence of derivations. *Journal of Automated Reasoning*, 2011. To appear.

[Cre08a] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proceedings of the 20th International Conference on Computer Aided Verification (CAV'08)*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[Cre08b] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proceedings of the 20th International Conference on Computer Aided Verification (CAV'08)*, Lecture Notes in Computer Science, 2008.

[CZ06] V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2006.

[DDMP04] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, 2004.

[DFGK09] A. Datta, J. Franklin, D. Garg, and D. Kaynar. A logic of secure systems and its application to trusted computing. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P'09)*, pages 221–236, 2009.

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

[Dij75] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[DJP10] N. Dong, H. Jonker, and J. Pang. Analysis of a receipt-free auction protocol in the applied pi calculus. In *Proceedings of the 7th International Workshop on Formal Aspects in Security and Trust (FAST'10)*, volume 6561 of *Lecture Notes in Computer Science*, pages 223–238, Pisa, Italy, 2010. Springer.

[DLM04] N. A. Durgin, P. Lincoln, and J. C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols (FMSP'99)*, Trento (Italy), 1999.

[DMRSar] A. Datta, J. Mitchell, A. Roy, and S. Stiller. *Formal Models and Techniques for Analyzing Security Protocols*, chapter Protocol Composition Logic. IOS Press, To appear.

[Dou02] J. R. Douceur. The sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

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

[DY81] D. Dolev and A. C. Yao. On the security of public key protocols. In *Proceedings of the 22nd Symposium on Foundations of Computer Science (FCS'81)*, pages 350–357, Nashville (Tennessee, USA), 1981. IEEE Computer Society Press.

[EG83] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Technical Report*. IEEE Computer Society Press, 1983.

[EMM07] S. Escobar, C. Meadows, and J. Meseguer. Maude-npa: Cryptographic protocol analysis modulo equational properties. In *Proceedings of the Foundations of Security Analysis and Design (FOSAD)*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2007.

[Eng85] J. Engelfriet. Determinacy implies (observation equivalence = trace equivalence). *Theoretical Computer Science*, 36:21–25, 1985.

[FGML09] T. Feng, X. Guo, J. Ma, and X. Li. UC-Secure Source Routing Protocol, 2009.

[FHF06] A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. `http://itpolicy.princeton.edu/voting/`, 2006.

[FOO92a] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology (AUSCRYPT'92)*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[FOO92b] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology – AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

[FRF+07] J. Freudiger, M. Raya, M. Félegyházi, P. Papadimitratos, and J.-P. Hubaux. Mix-zones for location privacy in vehicular networks. In *Proceedings of the ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS'07)*, 2007.

[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.

[GLNS93] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. In *IEEE Journal on Selected Areas in Communications, Vol.11, No.5, June, 1993, pp.648-656*, 1993.

[Goo10] D. Goodin. Defects in e-passports allow real-time tracking. The Register, 2010. `http://www.theregister.co.uk/2010/01/26/epassport_rfid_weakness/`.

[GRS+07] S. Gürgens, C. Rudolph, D. Scheuermann, M. Atts, and R. Plaga. Security evaluation of scenarios based on the TCG's TPM specification. In *Proceedings of the 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2007.

[GSS+07] Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan. Beyond secure channels. In *Scalable Trusted Computing (STC'07)*, pages 30–40, November 2007.

[GT00] J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proceedings of the 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Computer Society Press, 2000.

[HFP98] R. Housley, W. Ford, and W. Polk. X.509 certificate and CRL profile, 1998. IETF standard, RFC 2459.

[HPJ03] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the ACM Workshop on Wireless Security*, pages 30–40. ACM, 2003.

[HPJ05] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11:21–38, 2005.

[HPJ06] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006.

[Hut02] H. Huttel. Deciding framed bisimulation. In *Proceedings of the 4th International Workshop on Verification of Infinite State Systems INFINITY'02*, pages 1–20, 2002.

[JMB01] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[JVP10] M. Johansson, B. Victor, and J. Parrow. A fully abstract symbolic semantics for psi-calculi. In *Proceedings of the 6th Workshop on Structural Operational Semantics (SOS'09)*, EPTCS, pages 17–31, 2010.

[KK05] D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005.

[KMT11] S. Kremer, A. Mercier, and R. Treinen. Reducing equational theories for the decision of static equivalence. *Journal of Automated Reasoning*, 2011. To appear.

[KR05] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, (United Kingdom), 2005. Springer-Verlag.

[KRS10] S. Kremer, M. D. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *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, 2010. Springer.

[KS83] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In ACM, editor, *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 228 – 240, 1983.

[KSRW04] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Proceedings of the 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 27–28. IEEE Comp. Soc. Press, 2004.

[KSW97] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 1997.

[KT08] R. Küsters and M. Tuengerthal. Joint State Theorems for Public-Key Encryption and Digitial Signature Functionalities with Local Computation. In *Proceedings*

of the 21st IEEE Computer Security Foundations Symposium (CSF'08). IEEE Computer Society Press, 2008.

[KT09] R. Küsters and T. Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P'09)*, pages 251–266. IEEE Computer Society, 2009.

[KT11] R. Küsters and T. Truderung. Reducing protocol analysis with XOR to the XOR-free case in the Horn theory based approach. *Journal of Automated Reasoning*, 46(3-4):325–352, April 2011.

[LBD+04] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proceedings of Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer, 2004.

[LL10] J. Liu and H. Lin. A complete symbolic bisimulation for full applied pi calculus. In *Proceedings of the 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.

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Proceedings of 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322, Nara (Japan), 2005. Springer-Verlag.

[LLT06] P. Lafourcade, D. Lugiez, and R. Treinen. ACUNh: Unification and disunification using automata theory. In *Proc. 20th Int. Workshop on Unification (UNIF'06)*, pages 6–20, Seattle (Washington, USA), 2006.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, Berlin (Germany), 1996. Springer-Verlag.

[Low97] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW'97)*, pages 18–30, Rockport (Massachusetts, USA), 1997. IEEE Computer Society Press.

[Low99] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.

[Low04] G. Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1):83–98, 2004.

[LPM+05] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Wireless Communications and Networking Conference*, volume 2, pages 1193–1199 Vol. 2, 2005.

[MAFM02] S. Malladi, J. Alves-Foss, and S. Malladi. What are multi-protocol guessing attacks and how to prevent them. In *Proceedings of the 11th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2002)*, pages 77–82. lEEE Computer Society Press, 2002.

[McA93] D. McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2):284–303, 1993.

[Mea96] C. Meadows. Language generation and verification in the NRl protocol analyzer. In *Proceedings of the 9th Computer Seurity Foundation Workshop (CSFW'96)*. IEEE Computer Society Press, 1996.

[Mil09] J. K. Millen. Rewriting techniques in the constraint solver. *Electronic Notes in Theoretical Computer Science*, 234:77–91, 2009.

[Mit02] J. C. Mitchell. Multiset rewriting and security protocol analysis. In *Proceeding of the 13th International Conference on Rewriting Techniques and Applications (RTA'02)*, volume 2378 of *Lecture Notes in Computer Science*, pages 19–22, Copenhagen, Denmark, 2002. Springer.

[Möd10] S. Mödersheim. Abstraction by set-membership: Verifying security protocols and web services with databases. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*, pages 351–360, Chicago, Illinois, USA, October 2010. ACM Press.

[MS01] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175, 2001.

[MS03] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proceedings of the 16th Computer Security Foundation Workshop (CSFW'03)*, pages 47–62, Pacific Grove (California, USA), 2003. IEEE Computer Society Press.

[MS05] J. Millen and V. Shmatikov. Symbolic protocol analysis with an abelian group operator or Diffie-Hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.

[MVB10] S. Mödersheim, L. Viganò, and D. A. Basin. Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. *Journal of Computer Security*, 18(4):575–618, 2010.

[NH06] S. Nanz and C. Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1):203–227, 2006.

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

[Nut90] W. Nutt. Unification in monoidal theories. In *Proceedings of the 10th International Conference on Automated Deduction, (CADE'90)*, volume 449 of *Lecture Notes in Computer Science*, pages 618–632, Kaiserslautern (Germany), 1990. Springer-Verlag.

[Oka96] T. Okamoto. An electronic voting scheme. In *Proceedings of the IFIP World Conference on IT Tools*, pages 21–30, 1996.

[Oka97] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proceedings of the 5th International Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35. Springer, 1997.

[PH02] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.

[PPS+08] P. Papadimitratos, M. Poturalski, P. Schaller, P. Lafourcade, D. Basin, S. Capkun, and J.-P. Hubaux. Secure Neighborhood Discovery: A Fundamental Element for Mobile Ad Hoc Networking. *IEEE Communications Magazine*, 46(2):132–139, February 2008.

[RS03] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2003.

[RS05] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[RSA04] RSA Security Inc., v2.20. *PKCS #11: Cryptographic Token Interface Standard.*, June 2004.

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 1-3(299):451–475, 2003.

[SAF10] Safespot project. http://www.safespot-eu.org/, 2006-2010.

[SB10] B. Smyth and B. Blanchet. Automated verification of selected equivalences for synchronous cryptographic protocols. 2010.

[SC07] G. Steel and J. Courant. A formalism for parallel key search. In *Proceedings of the 3rd Workshop on Formal and Computational Cryptography (FCC'07)*, 2007.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[Sch96] S. Schneider. Security properties and CSP. In *Proceedings of the 17th Symposium on Security and Privacy (S&P'96)*, pages 174–187, Oakland (California, USA), 1996. IEEE Computer Society Press.

[Smy10] B. Smyth. ProSwapper, 2010. `http://www.bensmyth.com/proswapper.php`.

[SRC07] B. Smyth, M. Ryan, and L. Chen. Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In *Proceedings of the 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS'07)*, volume 4572 of *Lecture Notes in Computer Science*, pages 218–231, 2007.

[SS89] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8(1/2):51–99, 1989.

[SSBC09] P. Schaller, B. Schmidt, D. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Proceedings of the 22nd Computer Security Foundations Symposium (CSF'09)*. IEEE Computer Society Press, 2009.

[Sta06] I. Standard. IEEE standard. IEEE Trial-Use Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages, Approved 8 June 2006.

[Ste05] G. Steel. Deduction with XOR constraints in security API modelling. In *Proceedings of the 20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Computer Science*, pages 322–336, Tallinn (Estonia), 2005. Springer-Verlag.

[TD10] A. Tiu and J. E. Dawson. Automating open bisimulation checking for the spi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF'10)*, pages 307–321, Edinburgh, United Kingdom, 2010. IEEE Computer Society Press.

[TEB05] F. L. Tiplea, C. Enea, and C. V. Birjoveanu. Decidability and complexity results for security protocols. In *Proceedings of the 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.

[Tru07] Trusted Computing Group. TPM Specification version 1.2. Parts 1–3, revision 103. `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, 2007.

[Tsa07] E. Tsalapati. Analysis of PKCS#11 using AVISPA tools. Master's thesis, University of Edinburgh, 2007.

[Tur06] M. Turuani. The cl-atse protocol analyser. In *Proceedings of the 17th International Conference on Term Rewriting and Applications (RTA'06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Springer, 2006.

[vDR09] T. van Deursen and S. Radomirovic. Algebraic attacks on rfid protocols. In *Proceedings of the 3rd International Workshop on Information Security Theory and Practice (WISTP'09).*, volume 5746 of *Lecture Notes in Computer Science*, pages 38–51. Springer, 2009.

[Ver03] K. N. Verma. Two-way equational tree automata for AC-like theories: Decidability and closure properties. In *Proceedings of the 14th International Conference on*

Rewriting Techniques and Applications (RTA'03), volume 2706 of *Lecture Notes in Computer Science*, pages 180–196, Valencia (Spain), 2003. Springer-Verlag.

[VSS05] K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational Horn clauses. In *Proceedings of the 20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Computer Science*, pages 337–352, Tallinn (Estonia), 2005. Springer-Verlag.

[Wei99] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Proceedings of the 16th International Conference on Automated Deduction (CADE'99)*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 1999.

[YAB+05] P. Youn, B. Adida, M. Bond, J. Clulow, J. Herzog, A. Lin, R. Rivest, and R. Anderson. Robbing the bank with a theorem prover. Technical Report UCAM-CL-TR-644, University of Cambridge, August 2005.

[YB03] S. Yang and J. S. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03)*, 2003.

[You04] P. Youn. The analysis of cryptographic APIs using the theorem prover Otter. Master's thesis, Massachusetts Institute of Technology, 2004.