Département de formation doctorale en informatique
UFR STMIA

École doctorale IAEM Lorraine

Sécurité des protocoles cryptographiques: décidabilité et résultats de transfert

THÈSE

présentée et soutenue publiquement le 17 décembre 2007

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Eugen Zălinescu

Composition du jury

Président :	Yassine Lakhnech	Université Joseph Fourier, Grenoble
Rapporteurs :	Jean Goubault-Larrecq	École Normale Supérieure de Cachan
	Thomas Wilke	Université Christian-Albrechts, Kiel
Directeurs :	Véronique Cortier	CNRS, Nancy
	Michaël Rusinowitch	INRIA, Nancy
Examinateurs :	Philippe Even	Université Henri Poincaré, Nancy
	Cédric Fournet	Microsoft Research, Cambridge

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

English title: *Security of cryptographic protocols: decidability and transfer resultats*

Abstract

This thesis is developed in the framework of the symbolic analysis of security protocols. The contributions are represented by decidability and transfer results in the following directions which are major topics in protocol verification:

- treatment of the cryptographic primitives: CBC encryption, blind signatures;
- security properties: strong secrecy, existence of key cycles;
- approaches for protocol security: construction of the secure protocols.

Thus, we showed the decidability (on the one hand) of the existence of key cycles for a bounded number of sessions using a generalised constraint system approach, and (on the other hand) of secrecy for protocols using the CBC encryption or blind signatures for an unbounded number of sessions by using a refined resolution strategy on a new fragment of Horn clauses.

We also transferred protocol security from a weak framework towards a stronger framework in the following directions. On the one hand, we showed that a weak property of secrecy (i.e. reachability-based secrecy) implies under certain well-motivated assumptions a stronger secrecy property (i.e. equivalence-based secrecy). On the other hand, we built protocols secure against active adversaries considering an unbounded number of sessions, by transforming protocols which are secure in a non-adversarial setting.

Keywords: security protocols, decision procedures, CBC encryption, blind signatures, key cycles, strong secrecy, constraint systems, Horn clauses, applied pi calculus.

Résumé

Cette thèse se situe dans le cadre de l'analyse symbolique des protocoles Les contributions sont représentées par l'obtention de résultats de décidabilité et de transfert dans les directions suivantes qui sont des thèmes majeurs en vérification des protocoles :

- traitement des primitives cryptographiques : chiffrement CBC, signatures en aveugle;
- propriétés de sécurité : secret fort, existence de cycles de clefs;
- approches pour la sécurité : construction de protocoles sûrs.

Ainsi, nous avons montré la décidabilité (d'une part) de l'existence de cycles de clefs et (d'autre part) du secret pour des protocoles utilisant le mode de chiffrement CBC ou des signatures en aveugle.

Nous avons aussi transféré la sécurité des protocoles d'un cadre faible vers un cadre plus fort dans les sens suivants. D'une part, nous avons montré qu'une propriété de secret faible implique sous certaines hypothèses une propriété de secret plus forte. D'une autre part, nous avons construit des protocoles sûrs à partir de protocoles ayant des propriétés plus faibles.

Mots-clés: protocoles de sécurité, procédures de décision, chiffrement CBC, signatures en aveugles, cycles de clefs, secret fort, systèmes de contraintes, clauses de Horn, pi-calcul appliqué.

Acknowledgements

First of all, a big and warm thank-you to Véronique Cortier. I am grateful for the confidence she has shown to me, for her generous and constant availability, and for her clear and precise answers to my questions. I cordially thank Michael Rusinowitch for his encouragement, especially during the initial stages of the writing of this thesis and, in general, for his valuable advice, and for having offered a good atmosphere for the development of the thesis.

I also kindly thank the members of my thesis committee for having accepted to be on it. I particularly thank the reviewers, Jean-Goubault Larrecq, for his careful reading of the manuscript and for his pertinent remarks, and Thomas Wilke, for his observation that led to an alternative treatment of complexity in Chapter 2 (see §2.2.6), and for having bravely endured a presentation in a language that was not his own.

Warm thanks to Bogdan Warinschi for his support and optimism, and for the many discussions we had during his post-doc at Nancy. I also thank R. Ramanujam who welcomed me in his team at IMSC, Chennai.

I thank Cornelius Croitoru and Dorel Lucanu, professors at the Faculty of Computer Science of Iaşi, for helping me to continue my studies in France, and Philippe Langevin, supervisor of my masters' internship, who encouraged and helped me to pursue a PhD thesis.

I thank Cédric Fournet for accepting me as a post-doc in his team at the Microsoft Research-INRIA Joint Centre, and my collaborators there, Ricardo Corin and Karthik Bhargavan, for their patience and understanding during the final phase of the writing of this document.

Finally, I thank my friends and my family for their precious support during these three years.

Contents

Introduction

1	Cryptographic protocols		11
	1.1	Communication protocols. Terminology	11
	1.2	The intruder	12
	1.3	Security properties	13
	1.4	Cryptographic primitives	13
	1.5	Attacks	15
2	Analysis of cryptographic protocols		16
	2.1	Symbolic verification of security protocols	18
	2.2	Linking the symbolic and cryptographic approaches	21
	2.3	Decidability and transfer results	22
3	Contributions and plan of the thesis		22
	3.1	Part I. Decidability results	23
	3.2	Part II. Transfer results	24

Chapter 1
Models for cryptographic protocols

1.1	Preliminaries		28
	1.1.1	Terms over order-sorted signatures	28
	1.1.2	Positions and subterms	29
	1.1.3	Substitutions	31
	1.1.4	Equational theories and rewriting systems	33
	1.1.5	Term deduction systems	34
1.2	Cryptographic primitives and messages		34
	1.2.1	A sort system for cryptographic protocols	36
	1.2.2	A signature for cryptographic protocols	37
	1.2.3	Two deduction systems for cryptographic protocols	38
	1.2.4	On the use of a third deduction system	42

1.2.5 The deduction problem . . . 43
1.3 Roles . . . 44
1.3.1 Specification of roles . . . 44
1.3.2 Execution of roles . . . 44
1.3.3 Executable roles . . . 46
1.3.4 Roles with matching and roles with equality tests . . . 47
1.4 Protocols . . . 48
1.4.1 Specification of protocols . . . 48
1.4.2 Execution of protocols . . . 48
1.4.3 Executable protocols . . . 50

Part I Decidability results **53**

Chapter 2
Decidability results using constraint systems

2.1 The model . . . 56
2.1.1 Constraint systems . . . 56
2.1.2 From protocols to constraint systems . . . 57
2.1.3 Security properties . . . 58
2.2 Simplifying constraint systems . . . 60
2.2.1 Simplification rules . . . 60
2.2.2 Decision procedure in NP-time . . . 61
2.2.3 Correctness . . . 63
2.2.4 Completeness . . . 65
2.2.5 Termination in polynomial time . . . 68
2.2.6 An alternative approach to polynomial-time termination . . . 71
2.3 Decidability of some specialised security properties . . . 73
2.3.1 Detection of key cycles . . . 73
2.3.2 Secrecy for protocols with timestamps . . . 82
2.4 Conclusions . . . 84

Chapter 3
Decidability results for Horn clauses

3.1 The model . . . 85
3.1.1 Horn clauses . . . 86
3.1.2 From protocols to Horn clauses . . . 87

3.2 A fragment of Horn clauses . . . 90
3.2.1 Intruder clauses . . . 90
3.2.2 Protocol clauses . . . 91
3.2.3 Extending the intruder power . . . 91
3.3 A decidability result . . . 93
3.3.1 Ordered resolution . . . 93
3.3.2 Our resolution method . . . 94
3.3.3 A decidable class . . . 95
3.3.4 Examples . . . 97
3.3.5 Proofs of intermediate results . . . 98
3.4 Application to the Needham-Schroeder symmetric key protocol . . . 102
3.4.1 Presentation of the protocol . . . 102
3.4.2 Correcting the protocol . . . 103
3.4.3 A transformation preserving secrecy . . . 104
3.4.4 Secrecy of the corrected protocol . . . 106
3.5 Conclusions . . . 107

Part II Transfer results **109**

Chapter 4
From simple secrecy to strong secrecy

4.1 The model . . . 113
4.1.1 The applied pi calculus . . . 113
4.1.2 Modeling protocols within the applied pi calculus . . . 115
4.1.3 Secrecy properties . . . 118
4.2 Passive case . . . 120
4.2.1 Simple secrecy implies strong secrecy . . . 120
4.2.2 Generalisation of well-formed frames . . . 122
4.3 Active case . . . 126
4.3.1 Our hypotheses . . . 126
4.3.2 Main result . . . 130
4.3.3 Proofs of intermediate results . . . 132
4.4 Application to some cryptographic protocols . . . 135
4.4.1 Yahalom . . . 135
4.4.2 Needham-Schroeder symmetric key protocol . . . 135
4.4.3 Wide Mouthed Frog Protocol (modified) . . . 136

4.5 Conclusions 137

Chapter 5
A transformation for obtaining secure protocols

5.1 Comparison with Katz and Yung's compiler 141
5.2 The model 142
5.3 Security properties 144
5.3.1 A logic for security properties 144
5.3.2 Examples of security properties 146
5.4 Transformation of protocols 147
5.5 Transfer result 148
5.5.1 Honest, single session traces 148
5.5.2 Transferable security properties 149
5.5.3 Transference theorem 149
5.5.4 Honest executions 150
5.5.5 Proof sketch of the transference theorem 151
5.5.6 Detailed proofs 152
5.6 Conclusions 157

Conclusions and perspectives

Bibliography **163**

Introduction

Communication protocols are ubiquitous nowadays, being essential for the correct functioning of a wide range of applications involving electronic communicating devices. They are thus present in our now common activities, like talking on the mobile phone, chatting and emailing, watching cable TV, or shopping on the internet. In many such applications security is of primary concern.We want our communications to be private, our data to be unmodified during its transmission, to be sure of the identity of our communication partner.

Security protocols are then designed to ensure such goals, and they use cryptography to obtain the basic building bricks. However, even if these bricks are perfectly secure, the way they are combined in order to obtain a protocol is very important. Indeed, many protocols which were believed to be correct were later found to have flaws (not at all related to cryptanalysis). These flaws can thus be used by malicious entities, and can lead to major negative consequences once the protocol is already deployed, as the same flaw can be used over and over again until a patch is released. It is hence particularly important to perform careful analyses of security protocols in order to be sure that they do achieve the goals they are designed for.

1 Cryptographic protocols

1.1 Communication protocols. Terminology

A simple communication protocol is the one used when two people are meeting for the first time, and may be described as follows:

$$\begin{aligned} A \Rightarrow B: &\quad \text{“Hello, I am } A\text{.”} \\ B \Rightarrow A: &\quad \text{“I am } B\text{. Nice to meet you.”} \end{aligned}$$

We see that a protocol is a sequence of *rules*, each one specifying the *sender* (A in the first rule), the *receiver* (B in the first rule), and the *message* sent by the sender. In a protocol each participant *plays* a certain *role*. Here there are two: the *initiator* A and the *responder* B. The symbols A and B (abbreviations for Alice and Bob) in the right hand side of rules are generic names which denote the identities of the initiator and the responder respectively. In specific situations we need to *instantiate* this description (that is, the generic parts) to obtain the actual sequence of exchanged messages, thus talking about a *session* of the protocol. We may also instantiate only some role thus obtaining a *role session*. Participants, either generic or concrete, are also called *agents*, *principals*, or *parties*.

For example the role of Bob may be played by b, where b is some agent identity. Nothing prevents b from playing, in another session, the role of A. Moreover these sessions could be run *concurrently*, in other words their rules may be *interleaved*. For example, the following *execution* is possible:

$$\begin{array}{lll} (1).1 & a(A) \Rightarrow b(B): & \text{“Hello, I am } a\text{.”} \\ (2).1 & b(A) \Rightarrow c(B): & \text{“Hello, I am } b\text{.”} \\ (2).2 & c(B) \Rightarrow b(A): & \text{“I am } c\text{. Nice to meet you.”} \\ (1).2 & b(B) \Rightarrow a(A): & \text{“I am } b\text{. Nice to meet you.”} \end{array}$$

Here the numbers in parenthesis denote the session, and the numbers that follow denote the index of the rule within a session. Also, $b(A)$ denotes that the participant b is playing the role of Alice. A *run* of the protocol is the execution of a single session.

Each protocol is designed to achieve a certain *goal*. In the example above the goal is that the participants introduce themselves. The goal can be expressed by one or more *properties* the executions of the protocol should satisfy. Properties are generally dependent on the *environment* in which the protocol is deployed. Suppose, using the same example, that the participants want to have a confidential conversation. If their conversation takes place in a public place or over the telephone, the communication is clearly unsafe, since a *malicious* entity can listen to the conversations (without the participants even noticing it).

The situation is much the same when protocols are deployed over computer networks, where the endpoints are programs (or computers). Consider for example the Simple Mail Transfer Protocol, which can be schematically[1] described by:

$$\begin{array}{ll} A \Rightarrow S: & \text{“mail from:”}, A, \text{“rcpt to:”}, B, \text{“data”}, msg, \text{“.”} \\ S \Rightarrow B: & \text{msg} \end{array}$$

Here ‘,’ denotes the *concatenation* of messages and S denotes the role of the mail server. A user A simply specifies the sender (himself), the intended recipient B, and the content msg of the email. The main goal of this protocol is to send mail over the Internet, its *correctness* being formulated with respect to this requirement. However we see that the content of a mail is not protected from disclosure or tampering, and the participants need not be the ones they pretend to be (e.g. a can send a message starting with “mail from: c” instead of “mail from: a”). Indeed, malicious actions, like eavesdropping, tampering with, or forging messages, can be easily performed by a corrupted server, a malicious agent, or a packet sniffer. Hence it is desirable to ensure properties that show the impossibility of such actions. Such properties which rely on the existence of a malicious environment are called *security properties* and protocols which aim at guaranteeing them are *security protocols*.

1.2 The intruder

Security properties are particularly important mainly when the environment is unsafe. Hence when talking about security protocols we always assume a malicious environment. Concretely this environment takes the form of an agent with special capabilities, called the *intruder* and denoted I, also known as *adversary*, *attacker*, or *penetrator*. It is assumed that he can listen to the communication and hence knows all the messages that were sent on the network. If his capabilities are restricted to this one, we talk about a *passive* intruder. An *active* intruder can do much more. R. Needham and M. Schroeder [NS78] first described the capabilities of an active intruder:

> We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material.

[1] This description is approximate since each of the three parts of the first message are in fact sent sequentially and are followed by acknowledgements of the server; also, the second rule is not part of the protocol itself.

An intruder (being an agent) can play a role in the protocol, but he need not follow the rules of the protocol, as *honest agents* do. Moreover, he knows all the private data of *corrupted agents*, thus being able to play their roles without the others agent's notice. Also it is assumed that dishonest agents (i.e. that do not follow the protocol) are part of the malicious environment and are hence represented by the intruder. In other words, dishonest agent and corrupted agents are the same concept.

1.3 Security properties

Secrecy and authentication are basic properties that are required in many generic applications. Some specific applications however need properties tailored to their needs. For example, in contract-signing protocols we may ask for properties like fairness and non-repudiation, while in voting protocols anonymity (of the voters) and coercion-resistance are needed. Specifying and analysing these properties may also require dedicated techniques (for example from game theory).

Secrecy This property usually specifies that some messages should be known only by some agents, in particular they should not be known by the intruder. However we sometimes also require that the intruder is not able to infer anything about the secret messages. This is equivalent to saying that the intruder is not able to distinguish between executions of the protocol in which the secrets were replaced by arbitrary messages. To differentiate between the two versions of secrecy we call the former one *simple secrecy* and the latter one *strong secrecy*. Still another variant is when some values are to remain secret after revealing some other secret values, a property known as *forward secrecy*.

Authentication This property holds if the agents have proven their identity (to some other agents) in some way. Depending on the mechanism used to achieve it and/or on how much assurance is needed there can be many variants of this property. For example, it could be stated in an absolute way: the agents are right about the identities of their communication partners; or, depending on the mechanism that is used for authentication: the agents agree on some values (what was sent is what was received).

We have mentioned that the intruder has control over the communication, in particular he knows and is able to modify the messages which are sent over the network. Hence security properties could not be satisfied if we didn't have tools for ensuring the confidentiality and integrity of messages. Fortunately such tools exist, being provided and guaranteed by *cryptography*. Security protocols are thus also called *cryptographic protocols*.

1.4 Cryptographic primitives

Cryptographic tools have existed from ancient times, serving mostly military purposes, but it is only with the advent of electronic devices that cryptography has become an established and general purpose field (see, e.g., [MVO96, Sch93] for introductory texts).

Cryptographic primitives are the basic operations from which security is built. They operate on *bitstrings*. The most used operations are *encryption*, which provides message confidentiality, *hashing*, which assures message integrity, and *digital signing*, which provides message origin authentication.

Encryption hides information, while decryption reveals it. These operations are parameterised by *keys* which allow the same schema to be used by different parties. The information in clear used as input to the encryption algorithm is called *plaintext*, while the encrypted information (i.e. the output) is called *ciphertext*.

Symmetric encryption In such encryption schemes the same key is used to encrypt and to decrypt a message. Hence two parties share a key in order to be able to communicate securely, thus the alternative name *shared-key* encryption. The symmetric encryption of the plaintext M with the key K is denoted $\{\!\{M\}\!\}_K$.

Encryption of a message is usually done by cutting the message into several blocks of fixed-length and then using a block cipher (like DES or the more recent AES). The *encryption mode* specifies the way the block cipher is used to obtain the ciphertext. The simplest mode, called ECB (electronic codebook), operates by encrypting each block independently, the ciphertext being the concatenation of the results, that is the encryption of the message block sequence $P_1P_2\cdots P_n$ (where some bits may be added to P_n such that every block has the same length) with the key K is $\{\!\{P_1\}\!\}_K\{\!\{P_2\}\!\}_K\cdots\{\!\{P_n\}\!\}_K$.

In other modes, like the CBC (cipher-block chaining) mode, the encryption of a block depends on the encryption of the previous block. In the CBC mode (illustrated in Figure 1), the encryption of $P_1P_2\cdots P_n$ with K is $C_1C_2\cdots C_n$ where $C_0 = IV$ (initialisation vector) and $C_i = \{\!\{C_{i-1} \oplus P_i\}\!\}_K$ for $i \geq 1$, with $\oplus$ being the exclusive or (XOR) operation on bits.

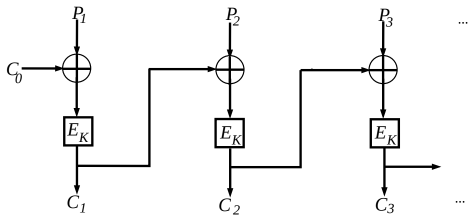


Figure 1: CBC encryption mode.

Encryption in the CBC mode has the following *prefix* property: if $C_1C_2\cdots C_iC_{i+1}\cdots C_n = \{\!\{P_1P_2\cdots P_iP_{i+1}\cdots P_n\}\!\}_K$ then $C_1C_2\cdots C_i = \{\!\{P_1P_2\cdots P_i\}\!\}_K$. That is to say that one (e.g. an intruder) can get $\{\!\{P\}\!\}_K$ from $\{\!\{P, P'\}\!\}_K$ if the length of P is a multiple of the block length used by the cryptographic algorithm. Note that encryption in the ECB mode also has the prefix property.

Asymmetric encryption In such encryption schemes, a.k.a. *public-key encryption schemes*, each user a has a pair of keys, the *public key* $\mathsf{ek}(a)$, used for encryption, and the *private key* $\mathsf{dk}(a)$, used for decryption. Public keys are made available to anyone, while private keys are known only by the owner. The encryption of a message M is this time denoted $\{\![M]\!\}_{\mathsf{ek}(a)}$. The security of public-key encryption relies on the difficulty to solve problems like integer factoring (as for the RSA system [RSA78]) or the discrete logarithm problem (as for the ElGamal system [Gam85]).

Symmetric encryption algorithms are several orders of magnitude faster than asymmetric ones. However, they are impractical in large networks due to the large number of keys needed to be exchanged a priori. Hence, the two systems are complementary and are frequently used together: public-key encryption is first used for establishing a *session key*, which is used in subsequent symmetric encryptions.

Digital signatures Digital signature schemes are used to bind a message with an entity: they compute a digital signature from the message and a private key of the entity (*signing key*). Given a signature and a *verification key*, which is public, one can check the authenticity of the signature. In other words, anyone can verify a signature but only the possessor of the signing key can sign.

In certain situations, for example in voting protocols, it is useful that a party signs messages without knowing them. This can be done using *blind signatures schemes*. For instance, such a scheme allows an agent (e.g. a voter) to have a message (e.g. a vote) signed blindly by an another entity (e.g. an administrator). In a typical implementation, the message is first *blinded* and then signed, to obtain a blind signature. Later on, the inverse operation of blinding, unblinding, can be applied on the blinded signature to obtain a valid signature on the initial message. These operations are illustrated in the Figure 2 (where m is the message, r is the "blinding key", and k is the signing key).

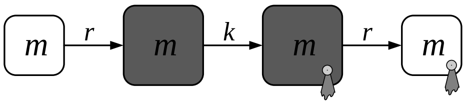


Figure 2: Blind signatures.

Hashing A *hash function* associates a (short) fixed length bitstring to an arbitrary length message. In cryptographic applications hash functions are one-way operations. Using hash functions, data integrity of a message can be verified easily, provided a hash of that message was securely stored. Indeed, it suffices to compute again the hash of the message and compare it with the stored one.

Cryptographic primitives may be *probabilistic* (or *deterministic*) depending whether some randomness is used (or not) when applying the primitive. Thus, when applied twice to the same inputs a probabilistic primitive gives two different outputs (except with negligible probability).

Besides cryptographic primitives, other basic elements present in security protocols are *nonces* and *timestamps*. Nonces are random numbers used no more than once for the same purpose, up to some negligible probability. Nonces and timestamps are intended to provide uniqueness or timeliness guarantees.

1.5 Attacks

A famous cryptographic protocol is the Needham-Schroeder public key protocol[2] [NS78]:

$$\begin{aligned} A \Rightarrow B: &\quad \{\!|N_a, A|\!\}_{\mathsf{ek}(B)} \\ B \Rightarrow A: &\quad \{\!|N_a, N_b|\!\}_{\mathsf{ek}(A)} \\ A \Rightarrow B: &\quad \{\!|N_b|\!\}_{\mathsf{ek}(B)} \end{aligned}$$

The goal of the protocol is the mutual authentication between A and B, meaning that if Bob has finished its run then he indeed played his role with Alice (as he believes he did), and symmetrically for Alice. She initiates a session by creating a fresh nonce N_a, concatenating it with her identity, encrypting the result with the public key of Bob and sending the encrypted message to him. Bob answers by copying the received nonce (the first component obtained after decrypting the

[2]Though over-cited, we believe that it still serves best as a pedagogical example.

received message with his private key), appending his own freshly generated nonce, encrypting the message with the public. Finally, Alice sends back to Bob his nonces encrypted with his public key. The role of nonces is to ensure authentication: only Bob can read the first message and find out the value of N_a, hence only he could have sent the second message. Similar reasoning applies for N_b.

Consider however the following execution:

$$
\begin{array}{lrcl}
(1).1 & A \Rightarrow I & : & \{\![N_a, A]\!\}_{\mathsf{ek}(I)} \\
(2).1 & I(A) \Rightarrow B & : & \{\![N_a, A]\!\}_{\mathsf{ek}(B)} \\
(2).2 & B \Rightarrow I(A) & : & \{\![N_a, N_b]\!\}_{\mathsf{ek}(A)} \\
(1).2 & I \Rightarrow A & : & \{\![N_a, N_b]\!\}_{\mathsf{ek}(A)} \\
(1).3 & A \Rightarrow I & : & \{\![N_b]\!\}_{\mathsf{ek}(I)} \\
(2).3 & I(A) \Rightarrow B & : & \{\![N_b]\!\}_{\mathsf{ek}(B)}
\end{array}
$$

Alice is starting a communication with a corrupted agent I (she is probably unaware of the agent being corrupted). The agent I is able to build the second message and to impersonate Alice to Bob. Thus Bob answers and his message is forwarded to Alice as coming from I (steps 3 and 4). Alice continues as expected (step 5), and again I impersonates Alice to Bob (step 6). Hence at the end of his run, Bob believes that he is talking with Alice while in fact he is talking with I. So the authentication of Alice to Bob doesn't hold.

The above description corresponds to an *attack*, that is a sequence of actions that the intruder performs in order to falsify a certain security property, that is to *break* it. Remark that the attack doesn't rely on the weakness of any of the cryptographic primitives, which are supposed to be secure, but on the *logical flaws* of the protocol (there is no information in the second message to deduce where the message comes from).

The above attack is an example of a *man-in-the-middle* attack. There are also other types of attacks, like:

- *replay* attacks, when the flaw is obtained by replaying some old messages (i.e. that have been sent previously, in the same or in other protocol runs);
- *type-flaw* attacks relying on a message of a certain type (e.g. an identity) being misinterpreted as a message of another type (e.g. a nonce);
- *guessing attacks* in which some secret message (e.g. a password) can be relatively easy guessed (because its set of possible values is small) and the guess can be checked for validity.

Coming back to the above protocol, the creators of the protocol were aware that it may be "prone to extremely subtle errors" and acknowledged that techniques to verify the correctness of security protocols are strongly needed. Indeed even though man-in-the-middle attacks were known before, it took 17 years to find such errors, the mentioned attack being discovered by G. Lowe [Low96].

2 Analysis of cryptographic protocols

Development of protocols. Mainly two steps are performed before the deployment of a cryptographic protocol: the first one is the *design* of the protocol and the second one is the *validation* of the protocol. There is an implicit loop here, the design being refined until the protocol is finally considered secure. There is thus a tight dependency between these two steps.

The design is guided by the security goals the protocol should verify and by the context in which it is deployed. The context, which can be given by the network structure (e.g. private vs. public channels, cable vs. wireless networks), the number of participants, their architecture (e.g. servers vs. clients, programs vs. hardware devices) and so on, imposes a number of constraints, like efficiency or limited resources. Hence the constraints can vary greatly, and this is a reason why there are plenty of protocols for achieving the same security goals.

As we have seen in the above example, informal arguments are not enough to validate secure protocols. And it is not a singular example. Indeed many of the released security protocols have flaws whether it is about "toy" protocols (used for study in the academic community) e.g. [Low96, CJ97, Spo], or about "real" protocols (used by the industry) as in [CJT$^+$06]. It is hence clear that rigorous analysis methods are mandatory to validate security protocols. Moreover, due to the large number of protocols and variants of them, and also due to the complexity of their analysis, (full or at least partial) automation is also a strong desiderata, both during the design process and after their release.

Two "worlds" for verification. For about 20 years (from late 70's to late 90's) two distinct and seemingly unrelated approaches have been used for rigorous validation of protocols. The models that these approaches use are called on the one hand *symbolic models* (a.k.a. Dolev-Yao, formal or abstract models), and the other hand *cryptographic models* (a.k.a. probabilistic, computational or concrete models). In the symbolic models, messages are modeled by elements (or equivalence classes) in a term algebra that the adversary can manipulate using a fixed set of symbolic operations. Thus these models introduce abstractions, which allow simpler reasoning about security of protocols, but are subject to questions about their faithfulness with respect to reality. In the cryptographic models, messages are bit strings and the adversary is an arbitrary probabilistic polynomial-time Turing machine. Being close to reality, results in these models yield strong security guarantees, but the validation proofs are often quite involved and only rarely suitable for automation (see e.g. [GM84, BR93]). It is only recently that automatic tools have appeared [BL06, Bla07] for cryptographic models.

The symbolic approach From now on we focus on the symbolic "world" (although references to the other world may occur).

Following previous work of R. Needham and M. Schroeder [NS78], D. Dolev and A. Yao [DY83] performed the first analysis in a symbolic model, hence the alternative name of this approach. A very important implicit abstraction introduced by their work is that encryption is perfect in the sense that no (not even partial) information about the plaintext can be obtained from a ciphertext without knowing the decryption key. When generalising this hypothesis to arbitrary cryptographic primitives we talk about the *perfect cryptography hypothesis*. Subsequently, S. Even and O. Goldreich [EG83] showed that secrecy is undecidable (even for protocols without nonces). This showed that the analysis is indeed a difficult problem, and that further abstractions or restrictions need to be formulated to tackle the problem.

Starting from these seminal works a new topic has emerged: the *symbolic verification of security protocols*, with the objectives: rigorous, automatic and faithful analysis of protocols. The results can be classified in number of ways: chronologically, by the class of protocols, by the set of primitives, by the type of attack or by the security property under study, by the aim of the analysis, by the model or by the method used in analysis, by the level of automation, etc. We will try in the following to sketch some of these criteria, focusing only on some of them.

2.1 Symbolic verification of security protocols

Symbolic approaches mainly focused, as we do, on key exchange and authentication protocols. However, as the applications of protocols have diversified and the verification methods have become more mature, voting protocols [DKR06], contract-signing protocols [KKW05], recursive protocols [KKW07], web-services protocols [BFG04, CLR07] etc. are being analysed using symbolic methods.

2.1.1 Security properties

A first difficulty of symbolic verification is to formally express the security properties that are expected. As we have seen, even a basic property such as secrecy admits two different acceptable definitions, namely reachability-based (simple) secrecy and equivalence-based (strong) secrecy, and these notions seemed unrelated [Aba00]. However, a quite surprising result (see [CW05]) states that the cryptographic counterparts of the two notions (simple secrecy can be translated into a similar reachability-based secrecy notion, and strong secrecy is close to indistinguishability—a standard security definition in cryptography) are related: cryptographic simple secrecy actually implies indistinguishability in the cryptographic setting.

Authentication has even more variants. They are often formulated by means of *corresponding assertions* [WL94]. G. Lowe has given a hierarchy of formulations [Low97], going from *aliveness* (which only requires that, when the authenticating agent finishes a run, the authenticated agent has at least participated in some run) to *injective agreement* which requires that to each run of the authenticating agent there is a unique corresponding run of the authenticated agent such that the two agents agree on some values.

Simple secrecy and authentication properties are usually expressed by predicates on *traces* (sequences of states and/or actions describing the execution of the system composed by a protocol and its environment), which have been intensively studied in the context of concurrent systems (but without considering security). Nevertheless, many other properties, like strong secrecy, anonymity, fairness, non-repudiation are not trace properties. The techniques used to treat these properties are usually different, and more subtle and involved. Some of these properties have only recently received proper formal definitions (see [CDE05] for guessing attacks, [KKT07] for properties of contract-signing protocols, or [DKR06] for properties of voting protocols). We mainly focus in this thesis on trace properties and in the rest of this section on simple secrecy.

2.1.2 Primitives and their properties

While the set of studied primitives is rather standard (symmetric and/or asymmetric encryption, digital signatures), it is the amount of faithfulness in capturing their properties that varies. Many cryptographic functions admit simple algebraic properties. For example, concatenation is associative, encryption is homomorphic in ECB mode, has the prefix property in CBC mode etc. In standard Dolev-Yao models, which assume the perfect cryptography assumption, these properties are ignored. For example, concatenation is modeled by *pairing* (denoted $\langle m, m' \rangle$) which is non-associative, that is $\langle m_1, \langle m_2, m_3 \rangle \rangle \neq \langle \langle m_1, m_2 \rangle, m_3 \rangle$. These algebraic properties can be exploited by intruders, and thus attacks may be missed if they are not taken into account. Moreover, such properties may be crucial too for a proper working of the protocol, as it is the case for some voting protocols which explicitly rely on the properties of blind signatures. Therefore, a lot of recent work has focused on weakening the perfect cryptography assumption, e.g. [AF01, CLS03, CD05, CR06].

2.1.3 Approaches

Security protocols are difficult to verify due to their infinite nature, given by several elements: the exchanged messages can have any size, they can use any number of new keys and nonces, the number of participants and sessions are not bounded. Indeed, focusing on reachability-based secrecy, several undecidability results show that these elements contribute to the difficulty of the problem. Thus, the problem remains undecidable even if one bounds the size of messages (see e.g. [DLMS99, AC02]), or the number of nonces generated during the execution of the protocol (see e.g. [CC05]). One then needs to find alternative approaches to the generic verification problem.

Search for attacks Since most attacks involve only a few messages and sessions, an approach is to first search for attacks, by considering only a subset of all the possible executions.

Indeed, most of the first automatic tools for protocol analysis were model-checkers (like FDR [DNL99], Murϕ [MMS97], or Brutus [CJM00]) which did discover many interesting attacks (see e.g. [Low96]). Such tools represent protocols as finite-state machines (and security properties by temporal logic formulas), usually by considering only messages of bounded size and a finite number of sessions. Another possibility for bounding the search space is to consider, as in [DLM04], messages of bounded size and a finite number of nonces, which leads to searching for the secret in a finite intruder knowledge.

Assuming a finite number of sessions, but no bound on the message size, the search space becomes again infinite. The standard way to approach this setting is to use "symbolic" techniques (which, intuitively, use symbolic states to represent sets of concrete states), as first suggested by the work of A. Huima in [Hui99]. The secrecy problem was then proved to be NP-complete in this setting by M. Rusinowitch and M. Turuani [RT01]. The same setting was formalised by J. Millen and V. Shmatikov in [MS01] as a constraint system problem (an attack is expressed as a sequence of constraints that the intruder should solve). To solve constraint systems, these are first transformed to simpler constraints, usually called solved forms, by using a small set of simplification rules (testing the satisfiability of these constraints is much more easy). Compared to [RT01], presenting the decision procedure using a small set of simplification rules makes it more easily amenable to further extensions and modifications. Indeed, constraint systems have become the standard model when considering a bounded number of sessions (see e.g. [CLS03, BCD07, DLLT07, CDL06] for results concerning algebraic properties developed within this framework.) The same approach is used to handle arbitrary trace properties in [CSE05, Cor06], and equivalence-based properties like strong secrecy and guessing attacks in [Bau05, Bau07]. Also, several tools [CE02, Tur06] have been developed for verifying protocols for a bounded number of sessions.

Proof of correctness Searching for attacks is an effective method, but it does not guarantee the correctness of a protocol. And, as we saw, the automatic verification of arbitrary protocols is not possible. Then, one can either renounce full automation, or use semi-decision procedures, or restrict the considered class of protocols, or still, perform some approximations (or consider combination of these possibilities).

Indeed, one of the first tools which does not restrict in any way the model is the NRL protocol analyser [Mea96] of C. Meadows. However, the user needs to interact with and "help" the tool in order to obtain an answer. In the same vein are the approaches which use theorem provers, like the inductive approach of L. Paulson [Pau98] which uses Isabelle to prove security properties.

If one does want full automation and no loss of generality then one needs to cope with semi-decision procedures which finish if there is an attack (and say so), but need not finish if the protocol is correct (and thus may fail to say so). In this category we find tools like Casrul [JRV00] or Athena [SBP01].

However, it is often the case that semi-decision procedures take too long before giving an answer (if they ever would). Then another way to proceed is to introduce approximations or abstractions in the model. These approximations need to be correct: if the protocol is proved secure using them then it is indeed secure (i.e. without them). The drawback is that approximations may introduce false attacks. An example of such analysis is the use of tree automata to recognise an over-approximation of the intruder knowledge, as it has been done in [Mon99, GK00, Gou00] or in the tool TA4SP [ABB+05]. Vice-versa, one can under-approximate the infinite set of "safe" messages as it has done in the Hermes tool [BLP03] by using a symbolic representation based on patterns. Still another example is the use of Horn clauses to represent (rules of) protocols. Horn clauses usually abstract away sessions and order of execution of rules, since they can be used any number of times. However, the advantage of this modeling is that one can then use efficient resolution strategies to search for proofs. This approach was pioneered by Ch. Weidenbach [Wei99], and it gave rise to an efficient tool, ProVerif [Bla01]. Moreover, reference implementations of protocols written in C or ML can be verified, by extracting a set of Horn clauses which is then passed on to tools like SPASS, `h1`, or ProVerif (see respectively [GP05] and [BFGT06]).

Even if the problem is undecidable in general one can still hope that it is decidable for a restricted (but still large) class of protocols. And indeed several such classes have been exhibited. A first decidability result was obtained in [DEK82] for the class of ping-pong protocols, protocols in which participants have no memory and can thus only apply some sequences of unary operators on the last received message and send the result back. This is however not a realist setting. Next, in the context of model-checking finite representations of protocols, G. Lowe showed in [Low99] that under strong restrictions on protocols this method is complete. Such restrictions imposed for example the absence of "blind copies"; a *blind copy* is the transfer by a participant of an unknown data from the received message to the sent message. In [CLC03a], this restriction was relaxed by allowing one blind copying, but the analysis only considered a finite number of nonces. Ramanujam and Suresh [RS03], considering again no blind copies, showed that for tagged protocols (protocols for which all encryptions in the specification contain a different tag, and which are thus distinguishable also in the execution) secrecy is decidable, even for protocols with nonces. A similar result [BP03b] is obtained for a simpler tagging scheme, but in the context of Horn clauses (which, as we mentioned, introduce approximations), showing that ProVerif always terminates for tagged protocols. All these decidability results show that a class of more "realistic" protocols for which the secrecy problem to be decidable could still be found.

Correctness by design A completely different approach is to avoid the verification problem, simply by designing from the start provably correct protocols. To our knowledge, in the cryptographic world this was not explored mainly because of the difficulty to produce proofs of correctness. Indeed, only a few protocols are having such proofs in a cryptographic setting [War05, BP03a, BCJ+06] (the situation is likely to change due to the development of automatic tools in this setting). However, a similar but slightly different approach is rather pervasive in cryptographic design: one starts with the design of a simple version of a system intended to work in restricted environments (i.e. with restricted adversaries) and then obtain, via a generic transformation, a more robust system intended to work in arbitrary environments. For example, Goldreich, Micali, and Wigderson show how to compile arbitrary protocols secure in the presence of participants that honestly follow the protocol (but may try to learn information they are not entitled to) into protocols secure in the presence of participants that may arbitrarily deviate from the protocol [GMW87]. Bellare, Canetti, and Krawczyk have shown how to transform a protocol that is secure when the communication between parties is authenticated into one that remains secure when this assumption is not met [BCK98].

In the symbolic world, few tools have been developed having as goal the automatic synthesis of secure protocols. For example, Perrig and Song [PS00] describe a tool which mainly works by exhaustively searching the protocol space and invoking Athena to test for the correctness of each generated protocol. However, due to the huge search space, the tool is limited to generate only three party protocols.

Symbolic approaches that focus on modular protocol design include the following ones. Datta, Derek, Mitchell, and Pavlovic [DDMP05] propose a framework for deriving security protocols from simple components such as nonces, certificates, encrypted or signed messages. Security properties are thus added to a protocol through generic transformations. M. Abadi, G. Gonthier, and C. Fournet [AFG02] give a compiler for programs written in a language with abstractions for secure channels into an implementation that uses cryptography with the aim to eliminate cryptographic security analysis in involved settings.

Finally, let us recall also that an often used technique is to patch flawed protocols and then to argue that the patched protocol is this time correct. Recently, this method has been automated in [LMH07].

2.2 Linking the symbolic and cryptographic approaches

As we have mentioned, two independent approaches have been developed for the analysis of security protocols. Nevertheless, in the late 90's these approaches have started to be related (see [PSW00, LMMS98, AR00] for some seminal works in this direction). For example, one particularly interesting path, opened by M. Abadi and P. Rogaway [AR00, AR02], consists in proving that the abstraction of cryptographic primitives made in the Dolev-Yao model is correct as soon as strong enough primitives are used in the implementation. The goal is to obtain the best of both worlds: relatively simple, automated security proofs that entail strong security guarantees. For example, in the case of asymmetric encryption, it has been shown [MW04a] that the perfect encryption assumption is a sound abstraction for encryption schemes satisfying the IND-CCA2 property, which corresponds to a very high and well-established security level.

However, it is not always sufficient to find the right cryptographic hypotheses. Symbolic models may need to be amended in order to be correct abstractions of the cryptographic models. This is in particular the case for symmetric encryption. For example, in [BP04], M. Backes and B. Pfitzmann consider extra-rules for the formal intruder in order to reflect the ability of a real intruder to choose its own keys in a particular manner. A more widely used requirement is to control how keys can encrypt other keys. In a passive setting, soundness results [AR02, MW04b] require that no *key cycles* can be generated during the execution of a protocol. Key cycles are messages like $\{\!\{k\}\!\}_k$ or $\{\!\{k_1\}\!\}_{k_2}, \{\!\{k_2\}\!\}_{k_1}$ where a key encrypts itself or more generally when the encryption relation between keys contains a cycle. Such key cycles have to be disallowed simply because usual security definitions for encryption schemes do not provide any guarantees when such key cycles occur. In the active setting, the typical hypotheses are even stronger. For instance, in [BP04, JLM05] the authors require that a key k never encrypts a key generated before k, or, more generally, that it is known in advance which key encrypts which other key. More precisely, the encryption relation has to be compatible with the order in which keys are generated, or more generally, it has to be compatible with an a priori given *order on keys*. We note that the absence of key cycles and related properties are not only trace properties but also message structure properties, and thus cannot be treated by standard techniques for trace properties.

2.3 Decidability and transfer results

We have already seen that one can attack the verification problem from different angles: either directly by searching for decidability results, or indirectly by transferring a problem from one setting to another setting in which the problem is solved or simpler. This was the case for transformations of protocols which are (in)secure in one setting to protocols which are secure in a stronger setting; or for soundness results of symbolic models with respect to cryptographic models. Let us mention one more such example.

There are many different models in which one reasons about security protocols, like process algebras (spi calculus, applied pi calculus, and variants), strand spaces, multiset rewriting, first-order logics etc. It is generally accepted that a characterisation of security protocols obtained in one model also holds in different models. For example, we say that the secrecy problem is NP-complete for a bounded number of sessions but we do not specify the model in which this was proved. However, only a few rigorous comparisons between models exist [CDL$^+$00, AB02, Bla05]. We can also see them as transfer results.

3 Contributions and plan of the thesis

In a phrase, the contributions of this thesis consist in improving the state of the art in the symbolic verification of cryptographic protocols while studying less explored features of in the following directions:

- cryptographic primitives: CBC encryption, digital blind signatures;
- security properties: strong secrecy, existence of key cycles;
- approaches to security: transferring security from weaker to stronger settings, transforming protocols.

These features have been studied before, but (at least at the beginning of this thesis) they represent(ed) a relative small fraction of the vast body of literature on cryptographic protocols which mainly focused on:

- cryptographic primitives: Dolev-Yao primitives (i.e. mainly pairing and perfect encryption);
- security properties: simple secrecy, authentication;
- approaches to security: direct verification of existing protocols.

We have thus also tackled (though not directly) two important related topics: weakening the perfect cryptography hypothesis (by considering the prefix property of encryption in CBC mode), and linking the symbolic and cryptographic approaches (by considering the existence of key cycles). Others, for example G. Bana [Ban05], S. Delaune [Del06], P. Lafourcade [Laf06], P. Adão [Adã06], R. Janvier [Jan06], L. Mazaré [Maz06], M.Baudet [Bau07], have recently focused directly on these topics in their theses.

Plan of the thesis After giving the necessary preliminary definitions, we present in Chapter 1 how security protocols are modeled. The particular model we choose is inspired from the symbolic model of D. Micciancio and B. Warinschi [MW04a] and is rather standard for modeling an unbounded number of sessions. It has the advantage of being intuitive and explicit (w.r.t. to the actions of the intruder and of other agents). Whenever we work in a different model we describe briefly its relationship with this reference model.

According to the classification given in the previous section we separate our contributions by the approach: a direct one (obtaining decidability results) and an indirect one (obtaining transfer results). Each main contribution is then presented in different chapters, as shown next.

3.1 Part I. Decidability results

3.1.1 Chapter 2. Deciding the existence of key cycles

A first contribution is an NP-complete decision procedure for detecting the generation of key cycles during the execution of a protocol, in the presence of an active intruder, for a bounded number of sessions. This procedure deals with several versions of the definition of key cycles (for example, key cycles à la Abadi-Rogaway, or key orders à la Backes). We therefore provide a necessary component for the approach which consists in proving security properties in the cryptographic world by starting from security proofs of these properties in the symbolic world (and using soundness results like the ones presented in Section 0.2.2 in order to achieve this).

We have obtained the decidability of key cycles by generalising the constraint system approach. Indeed, we use the same simplification rules as in [CLS03], but in addition we show that this method is applicable to any security property that can be expressed as a predicate on the protocol trace and the agent memories. Compared to [CLS03], the framework is also extended to more general primitives, since we consider sorted terms, symmetric and asymmetric encryption, pairing and signatures (but we do not consider algebraic properties). Moreover, we prove termination in *polynomial time* of the (non-deterministic) decision procedure. This establishes the complexity of the constraint system approach, and also of the problem under study (modulo its complexity on solved forms).

We further illustrate the applicability of our generic approach, by giving an alternative and simple proof of the co-NP-completeness of secrecy for protocols with timestamps. We actually retrieve a significant fragment of the decidable class identified by L. Bozga *et al* [BEL04].

3.1.2 Chapter 3. Deciding a fragment of Horn clauses for protocols with CBC encryption and blind signatures

We propose a resolution strategy for deciding a fragment of first-order logic that allows one to incorporate the prefix property of CBC encryption in our protocol modeling and to prove the absence of attacks exploiting this property. The same fragment applies to abstract properties of blind signature schemes. The approach follows the line of [CLC03a] but requires a refined strategy in order to eliminate the additional clauses generated by resolution due to the new properties. As a consequence, we obtain that secrecy of cryptographic protocols can be proven for an unbounded number of sessions, in the case for example of CBC encryption and blind signatures, when nonces are abstracted by constant terms and at most one blind copy is performed at each transition.

We apply the verification algorithm to Needham-Schroeder symmetric key protocol, which is subject to an attack when the CBC encryption mode is used [PQ00]. We show how to fix the protocol and we prove the correction of the resulting protocol. The latter is done automatically, as we have extended a prototype implementation of the procedure in [CLC03a].

3.2 Part II. Transfer results

3.2.1 Chapter 4. From simple to strong secrecy

Motivated by the result of [CW05] and the large number of available systems for simple secrecy verification, we initiate a systematic investigation of situations where simple secrecy entails strong secrecy. This happens in many interesting cases.

We offer results in both passive and active cases in the setting of the applied pi calculus [AF01]. We first treat the case of passive adversaries. We prove that simple secrecy implies strong secrecy, as long as probabilistic primitives are used, and if the secret is not used to encrypt messages. The former condition is not a restriction since probabilistic encryption is *de facto* the standard in almost all cryptographic applications. The latter hypothesis is sustained by counter-examples. Next, we consider the more challenging case of active adversaries. We give sufficient syntactic conditions on the protocols for simple secrecy to imply strong secrecy. Intuitively, we require in addition that the conditional tests are not performed directly on the secret since such tests may provide information on the value of this secret. We again exhibit several counter-examples to motivate the introduction of our conditions. An important aspect of our result is that we do not make any assumption on the number of sessions: we put no restriction on the use of replication. In particular, our result holds for an unbounded number of sessions.

The interest of this contribution is twofold. First, conceptually, it helps to understand when the two definitions of secrecy are actually equivalent. Second, we can transfer many existing results (and the armada of automatic tools) developed for simple secrecy. For instance, since the simple secrecy problem is decidable for tagged protocols for an unbounded number of sessions [RS03], by translating the tagging assumption to the applied-pi calculus, we can derive a first decidability result for strong secrecy for an unbounded number of sessions for the class of protocols satisfying our conditions. Other decidable fragments might be derived from [DLMS99] for bounded messages (and nonces) and [AL00] for a bounded number of sessions. We exemplify our approach by showing strong secrecy of three protocols from the literature (starting from the known fact that these protocols satisfy simple secrecy).

3.2.2 Chapter 5. A transformation for obtaining secure protocols

Finally, we present a transformation that maps a protocol secure in an extremely weak sense (essentially in a model where no adversary is present) into a protocol that is secure against a fully active adversary which interacts with an unbounded number of protocol sessions. The transformation works for arbitrary protocols with any number of participants, written with usual cryptographic primitives. It provably preserves a large class of trace security properties that contains secrecy and authentication. Conceptually, the transformation is very simple, and has a clean, well motivated design. Each message is tied to the session for which it is intended via digital signatures and on-the-fly generated session identifiers, and prevents replay attacks by encrypting the messages under the recipient's public key.

The table on the next page shows a summary of properties, primitives, approaches, and models used in this thesis.

The contributions presented in Chapters 2, 3, 4, and 5 have been published in the Proceedings of LPAR'06 [CZ06], PPDP'05 [CRZ05], CSL'06 [CRZ06], and ESORICS'07 [CWZ07] conferences respectively. These contributions represent joint work with Véronique Cortier (in all papers), Michaël Rusinowitch (in the second and the third papers), and Bogdan Warinschi (in the fourth paper).

Ch.	Sec.	Properties	Primitives	Approach	Model
2	2.1, 2.2	trace properties	Dolev-Yao	search for attacks	constraint systems
	2.3.1	key cycles			
	2.3.2	secrecy (with timestamps)			
3		secrecy	CBC encryption blind signatures	proof of correctness	Horn clauses
4		(simple and strong) secrecy	Dolev-Yao	transfer	applied pi calculus
5		trace properties	Dolev-Yao	transfer	trace model [MW04a]

Table 1: Summary of properties, primitives, approaches, and models used in this thesis.

Chapter 1

Models for cryptographic protocols

Contents

1.1 Preliminaries . . . 28
- 1.1.1 Terms over order-sorted signatures . . . 28
- 1.1.2 Positions and subterms . . . 29
- 1.1.3 Substitutions . . . 31
- 1.1.4 Equational theories and rewriting systems . . . 33
- 1.1.5 Term deduction systems . . . 34

1.2 Cryptographic primitives and messages . . . 34
- 1.2.1 A sort system for cryptographic protocols . . . 36
- 1.2.2 A signature for cryptographic protocols . . . 37
- 1.2.3 Two deduction systems for cryptographic protocols . . . 38
- 1.2.4 On the use of a third deduction system . . . 42
- 1.2.5 The deduction problem . . . 43

1.3 Roles . . . 44
- 1.3.1 Specification of roles . . . 44
- 1.3.2 Execution of roles . . . 44
- 1.3.3 Executable roles . . . 46
- 1.3.4 Roles with matching and roles with equality tests . . . 47

1.4 Protocols . . . 48
- 1.4.1 Specification of protocols . . . 48
- 1.4.2 Execution of protocols . . . 48
- 1.4.3 Executable protocols . . . 50

As mentioned in the introduction we work in so called symbolic (or abstract, Dolev-Yao) models that represent messages by elements (or equivalence classes) in some term algebra. In this chapter we mainly present how protocols are modeled within this setting. We start by giving, in Section 1.1, the basic technical definitions and notions used throughout this document. We then present, in Section 1.2, how messages and the operations on them are represented. Next, in Sections 1.3 and 1.4, we show how we model protocols.

1.1 Preliminaries

This section mainly introduces the term algebra setting used in this thesis.

For a set S we denote by S^* the free monoid of words over S, by $\cdot$ the concatenation operator over S and ϵ the empty word. We may omit the symbol $\cdot$ when writing a word over S. The cardinality of a set S is denoted by $\sharp S$. Also we write 2^S for the power set of S. By infinite set we mean a countably infinite set (i.e. with the same cardinality as $\mathbb{N}$), and we say that a set is at most countable if it is finite or countably infinite. For a natural number n we write $[n]$ for the set $\{1, 2, \ldots, n\}$, with the convention that $[0] = \emptyset$. For a binary relation ρ we denote by ρ^+ its transitive closure and by ρ^* its reflexive and transitive closure.

1.1.1 Terms over order-sorted signatures

Let $(\mathsf{Sorts}, \leq)$ be a finite partially ordered set, its elements being called *basic sorts*. A *sort* is a pair $(w, \mathsf{s}) \in (\mathsf{Sorts}^* \times \mathsf{Sorts})$, denoted $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$ if $n \geq 1$ and simply s if $n = 0$, where $w = \mathsf{s}_1 \ldots \mathsf{s}_n$. By language abuse, we often call basic sorts just sorts. Furthermore, we assume that $(\mathsf{Sorts}, \leq)$ is a tree, where a tree is a partially ordered set such that for each $\mathsf{s} \in \mathsf{Sorts}$, the set $\{\mathsf{s}' \in \mathsf{Sorts} \mid \mathsf{s} \leq \mathsf{s}'\}$ is well-ordered by the relation $\geq$ (with $\mathsf{s} \geq \mathsf{s}'$ iff $\mathsf{s}' \leq \mathsf{s}$).

We consider an infinite set of *variables* $\mathcal{X}$, and an infinite set of *names* $\mathcal{N}$. Each variable and name has associated a unique basic sort, and for each sort there is an infinite number of variables and names of that sort. For a sort s, we denote by $\mathcal{X}_\mathsf{s}$ (and $\mathcal{N}_\mathsf{s}$) the set of variables (and respectively, names) of sort s.

Let $\mathcal{F}$ be an at most countable non-empty set of *function symbols*. For each function symbol f there is a unique associated sort $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$. This association is usually denoted by $f : \mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$, and n is called the *arity* of f. Function symbols of arity 0 are called *constants*. The set of function symbols of sort (w, s) is denoted by $\mathcal{F}_{w,\mathsf{s}}$.

The set $\mathcal{F}$ is also called a *signature*. An *order-sorted signature*[3] is a tuple $\Sigma = (\mathsf{Sorts}, \mathcal{F}, \leq)$, with $(\mathsf{Sorts}, \leq)$ and $\mathcal{F}$ as above.

A *term* over the signature $\mathcal{F}$ is defined inductively by:

- elements of $\mathcal{X} \cup \mathcal{N}$ are terms, and
- if $f \in \mathcal{F}$ has arity n and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term.

We denote by $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$ the set of terms over the signature $\mathcal{F}$. We say that a term *has sort* s, and we denote it $t : \mathsf{s}$, if

- $t \in \mathcal{X}_\mathsf{s} \cup \mathcal{N}_\mathsf{s}$, or
- $t = f(t_1, \ldots, t_n)$, $f \in \mathcal{F}_{\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}}$ and $t_1, \ldots t_n$ are terms of sorts $\mathsf{s}'_1, \ldots, \mathsf{s}'_n$ respectively with $\mathsf{s}'_i \leq \mathsf{s}_i$ for all $1 \leq i \leq n$.

Note that a term has at most one (basic) sort. Terms that do not have a sort are called *ill-sorted*. In contrast, terms that have a sort are called *well-sorted*. Remark also that when Sorts is a singleton every term is well-sorted. We denote by $\mathcal{T}_\mathsf{s}(\Sigma, \mathcal{X}, \mathcal{N})$ the set of terms of sort s, and let $\mathcal{T}(\Sigma, \mathcal{X}, \mathcal{N}) \stackrel{\text{def}}{=} \bigcup_{\mathsf{s} \in \mathsf{Sorts}} \mathcal{T}_\mathsf{s}(\Sigma, \mathcal{X}, \mathcal{N})$ be the set of well-sorted terms.

[3] This notion of order-sorted signature is a simplified version of what one usually finds in the literature (see, e.g. [GM92]), since here function symbols are not overloaded (i.e. they have a unique sort).

Exemple 1.1 *Consider* $\mathsf{Sorts} = \{\mathsf{A}, \mathsf{B}\}$ *with* $\mathsf{A} < \mathsf{B}$, $\mathcal{F} = \{f : \mathsf{A} \to \mathsf{A}, g : \mathsf{A} \to \mathsf{B}\}$, *and* x, y *variables of sort* A *and* B *respectively. Then* $f(x)$ *has sort* A, $g(x)$ *has sort* B *and* $f(y)$, $g(y)$ *are ill-sorted terms.*

Unless explicitly mentioned, we consider only well-sorted terms. Thus, and by abuse of notation, we usually denote an order-sorted signature Σ by its signature $\mathcal{F}$, especially when the set of sorts is clear from the context.

For a set of function symbols $F \subseteq \mathcal{F}$, a set of variables $X \subseteq \mathcal{X}$, and a set of names $N \subseteq \mathcal{N}$ we denote by $\mathcal{T}(F, X, N)$ the set of terms with function symbols in F, variables in X, and names in N. The sets $\mathcal{T}(F, X, \emptyset)$, $\mathcal{T}(F, \emptyset, N)$ and $\mathcal{T}(F, \emptyset, \emptyset)$ are abbreviated by $\mathcal{T}(F, X)$, $\mathcal{T}(F, N)$, and $\mathcal{T}(F)$ respectively. Moreover, we may use simply $\mathcal{T}$ instead of $\mathcal{T}(F, X, N)$ if the sets F, X, N are clear from the context. We denote the set of variables (names) occurring in a term t by $\mathrm{var}(t)$ (respectively $\mathrm{names}(t)$). A term without variables is called *ground* or *closed*. If T, T' are sets of terms and t, t' are terms we abbreviate $T \cup T'$ by T, T', $T \cup \{t\}$ by T, t, and $\{t, t'\}$ by t, t'.

1.1.2 Positions and subterms

We denote by $\mathbb{N}_+$ the set of positive integers. Then $\mathbb{N}_+^*$ is the set of sequences of positive integers. We call *positions* the elements of $\mathbb{N}_+^*$. We say that a position p is *smaller* than a position q, and we write $p \leq q$, if p is a prefix of q.

Given a term t, the set of positions of t, denoted by $\mathrm{pos}(t)$, is defined inductively as follows:

- if t is a variable or a name then $\mathrm{pos}(t) = \{\epsilon\}$;
- if $t = f(t_1, \ldots, t_n)$ then $\mathrm{pos}(t) = \{\epsilon\} \cup \bigcup_{1 \leq i \leq n} \{i \cdot p \mid p \in \mathrm{pos}(t_i)\}$.

Given a term t and a position $p \in \mathrm{pos}(t)$, the *subterm* of t at position p, denoted by $t|_p$, is defined inductively by:

- if $p = \epsilon$ then $t|_p \stackrel{\text{def}}{=} t$,
- if $p = i \cdot p'$ then $t|_p \stackrel{\text{def}}{=} t_i|_{p'}$, where $t = f(t_1, \ldots, t_n)$ for some $f \in \mathcal{F}$ and some terms $t_1, \ldots, t_n$.

A term u is a *(proper) subterm* of a term v iff there is a position $p \in \mathrm{pos}(v)$ such that $u = v|_p$ (and $u \neq v$). We extend the notion of subterm to sets of terms and say that a term u is a subterm of a set of terms T if u is a subterm of t for some $t \in T$. We write $\mathrm{st}(t)$ and $\mathrm{st}(T)$ for the set of subterms of a term t, and of a set of terms T, respectively. We denote by $\leq_{\mathrm{st}}$ ($<_{\mathrm{st}}$) the *subterm (strict) ordering*, with $u \leq_{\mathrm{st}} v$ ($u <_{\mathrm{st}} v$) iff u is a (proper) subterm of v.

The *head symbol* of a term t, denoted by $\mathrm{head}(t)$, is defined by $\mathrm{head} : \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N}) \to \mathcal{F} \cup \mathcal{X} \cup \mathcal{N}$, with

- $\mathrm{head}(t) = t$ if t is a name or a variable,
- $\mathrm{head}(t) = f$ if $t = f(t_1, \ldots, t_n)$.

An *occurrence* of a subterm t' in a term t is a position $p \in \mathrm{pos}(t)$ such that $t|_p = t'$. An occurrence of a function symbol f in a term t is a position $p \in \mathrm{pos}(t)$ such that the head symbol of $t|_p$ is f. Also we write $\mathrm{pos}_{\mathrm{v}}(t)$ for the set of variable positions of t (i.e. occurrences of variables in t), and $\mathrm{pos}_{\mathrm{nv}}(t)$ for the set of non-variable positions of t (i.e. occurrences of names and function symbols in t).

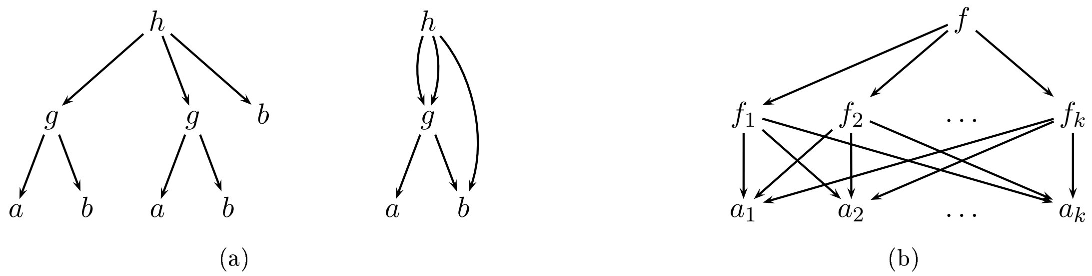


Figure 1.1: (a) The tree and the DAG representations of $t = h(g(a, b), g(a, b), b)$; (b) the DAG representation of $f(f_1(a_1, \dots, a_k), \dots, f_k(a_1, \dots, a_k))$.

For two terms u, v and a position $p \in \mathrm{pos}(t)$ such that $u|_p$ and v have the same sort, $u[v]_p$ denotes the term obtained by replacing in u the subterm at position p by v. Formally we have the following inductive definition:

- if $p = \epsilon$ then $u[v]_p \stackrel{\text{def}}{=} v$,
- if $p = i \cdot p'$ then $u = f(u_1, \dots, u_n)$ with $1 \leq i \leq n$, and

$$u[v]_p \stackrel{\text{def}}{=} f(u_1, \dots, u_{i-1}, u_i[v]_{p'}, u_{i+1}, \dots, u_n).$$

Representations of terms The *tree-representation* of a term t is the ordered directed tree $G = (V, E)$ with $V = \mathrm{pos}(t)$ and $E = \{(p, i, q) \mid p \cdot i = q\}$, where a triple (n, i, n') denotes the i-th outgoing arc from the parent node n (to the child node n'). Remark that the size of this representation is linear in the number of nodes of the tree.

The *size* of a term t is

$$|t| \stackrel{\text{def}}{=} \sharp\mathrm{pos}(t)$$

and the size of a set of terms T is $|T| \stackrel{\text{def}}{=} \sum_{t \in T} |t|$.

In a tree-representation of t, nodes p are in a many-to-one correspondence to subterms of t (i.e. $t|_p$). A more compact representation is obtained by considering a one-to-one correspondence between nodes and subterms (see Figure 1.1a).

The *DAG-representation* of a term t is the ordered DAG (directed acyclic graph) $G = (V, E)$ with $V = \mathrm{st}(t)$ and $E = \{(u, i, v) \mid u = f(u_1, \dots, u_n), v = u_i, 1 \leq i \leq n\}$. We observe that $\sharp E = \sum_{u \in \mathrm{st}(t)} k_u$, where k_u is the arity of the head function symbol of u. Thus $\sharp E \leq k \times \sharp(\,\mathrm{st}(t))$, where k is the maximal arity of function symbols in the signature (see Figure 1.1b). Supposing that the signature is fixed, and that DAGs are implemented with lists, the size of this representation is linear in the number of subterms. Given a set of terms T, a single DAG can represent all terms in T (consider this time $V = \mathrm{st}(T)$) provided that for each term there is pointer to the corresponding node in the graph. Observe that the number of terms in T, thus of pointers, is smaller than the number of subterms of T, thus the size of the representation of T is still linear in the number of subterms of T.

The *dag-size* of a term t is

$$|t|_{dag} \stackrel{\text{def}}{=} \sharp\mathrm{st}(t)$$

and the dag-size of a set of terms T is $|T|_{dag} \stackrel{\text{def}}{=} \sharp\mathrm{st}(T)$.

1.1.3 Substitutions

A *substitution* is a function $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$ from variables to terms. We say that a substitution σ *preserves sorts* if for any variable x, $\sigma(x)$ is well-sorted and if x has sort s and $\sigma(x)$ has sort s' then $\mathsf{s}' \leq \mathsf{s}$. Note that if u is well-sorted and σ is sort-preserving then $\sigma(u)$ is well-sorted. Moreover, if u is not a variable then u and $\sigma(u)$ have the same sort. Unless explicitly mentioned, we consider only sort-preserving substitutions.

Exemple 1.2 *Consider the sorts* $\mathsf{Nonce} < \mathsf{Msg}$*,* x, y *variables of sort* Nonce *and* Msg *respectively,* n *a name of sort* Nonce *and* t *a term of sort* Msg*. Then* σ_1 *and* σ_2 *given by* $\sigma_1(x) = n$*,* $\sigma_1(y) = t$*, and* $\sigma_2(y) = n$*, are sort-preserving substitutions, while* σ_3 *given by* $\sigma_3(x) = t$ *does not preserve sorts.*

For a substitution σ we define the domain of σ: $\mathrm{dom}(\sigma) \stackrel{\text{def}}{=} \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ and the range of σ: $\mathrm{ran}(\sigma) \stackrel{\text{def}}{=} \{\sigma(x) \mid x \in \mathrm{dom}(\sigma)\}$. The substitution σ with $\mathrm{dom}(\sigma) = \emptyset$ is called the *empty substitution* or the *identity substitution*. A substitution σ uniquely extends to a function $\overline{\sigma} : \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N}) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$ from terms to terms, defined inductively in the following way:

- if $t = x$ is a variable and $x \in \mathrm{dom}(\sigma)$ then $\overline{\sigma}(x) = \sigma(x)$,
- if $t = x$ is a variable and $x \notin \mathrm{dom}(\sigma)$ then $\overline{\sigma}(x) = x$,
- if $t = n$ is a name then $\overline{\sigma}(n) = n$,
- if $t = f(t_1, \ldots, t_n)$ then $\overline{\sigma}(t) = f(\overline{\sigma}(t_1), \ldots, \overline{\sigma}(t_n))$.

We overload the notation and denote this extension in the same way, i.e. σ instead of $\overline{\sigma}$, without loss of precision.

The application of a substitution σ to a term t is also written $t\sigma$. Similarly, for a set of terms T, $T\sigma$ denotes $\sigma(T)$, that is $\{t\sigma \mid t \in T\}$. A substitution is *ground* or *closed* if $\mathrm{var}(\sigma)$ is a set of ground terms. We denote $\mathrm{var}(\sigma) \stackrel{\text{def}}{=} \mathrm{var}(\mathrm{ran}(\sigma))$.

When the domain of a substitution σ is finite then we may denote it as $\sigma = \{{}^{t_1}/_{x_1}, \ldots, {}^{t_n}/_{x_n}\}$, where $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ and $\sigma(x_i) = t_i$ for all $1 \leq i \leq n$. If u and v are two terms then the term $u[{}^{v}/_{x}]$ is the term u where each occurrence of x has been replaced by v, that is $u[{}^{v}/_{x}] \stackrel{\text{def}}{=} u\sigma$ where $\sigma = \{{}^{v}/_{x}\}$.

A *renaming* ρ is a substitution with $\mathrm{ran}(\rho) \subseteq \mathcal{X}$ such that its restriction on $\mathrm{dom}(\rho)$ is a one-to-one function. If θ and σ are two substitutions then $\sigma\theta$ denotes their composition, that is $x(\sigma\theta) \stackrel{\text{def}}{=} (x\sigma)\theta = \theta(\sigma(x))$. It follows that $t(\sigma\theta) = (t\sigma)\theta$. A substitution θ is an *extension* of a substitution σ if $\mathrm{dom}(\sigma) \subseteq \mathrm{dom}(\theta)$ and $x\theta = x\sigma$, for all $x \in \mathrm{dom}(\sigma)$. A substitution σ is a *restriction* of a substitution σ if θ is an extension of σ. Given two substitutions σ and σ' with disjoint domains, the *union* of σ and σ', denoted $\sigma \uplus \sigma'$, is defined by $(\sigma \uplus \sigma')(x) = \sigma(x)$ if $x \in \mathrm{dom}(\sigma)$, and $(\sigma \uplus \sigma')(x) = \sigma'(x)$ if $x \in \mathrm{dom}(\sigma')$.

A substitution σ is *cyclic* if there exist $x_1, \ldots, x_n \in \mathrm{dom}(\sigma)$ with $n \geq 1$ such that $x_{i+1} \in \mathrm{var}(x_i\sigma)$ for all $1 \leq i \leq n$, with $x_{n+1} = x_1$. A substitution σ is *idempotent* if $\sigma = \sigma\sigma$, or, equivalently, if $\mathrm{var}(\mathrm{ran}(\sigma)) \cap \mathrm{dom}(\sigma) = \emptyset$. Note that idempotent substitutions are acyclic. We only consider acyclic substitutions.

If u and v are terms then a *unifier* of u and v is a substitution σ such that $u\sigma = v\sigma$. We say that u and v *unify* if they have a unifier.

For unsorted signatures, it is well-known (see, e.g. [FB01]) that if such a unifier exists then there exists a *most general unifier* θ, denoted by $\mathrm{mgu}(u, v)$, such that for all unifier σ there exists a substitution σ' such that $\sigma = \theta\sigma'$. The following proposition shows that this is also the case for the order-sorted signatures we consider here, where we say that θ is a *most general sort-preserving unifier* if for all sort-preserving unifier σ there exists a sort-preserving substitution σ' such that $\sigma = \theta\sigma'$.

Proposition 1.3 *If u and v are well-sorted terms that have a sort-preserving unifier then there exists a most general sort-preserving unifier of u and v.*

Proof Let θ be a (possibly not well-sorted) most general unifier of u and v.

For each $y \in \mathrm{var}(\theta)$, we define the set $S_y = \{\mathsf{s}_i \mid x \in \mathrm{dom}(\theta), p \in \mathrm{pos}(x\theta), x\theta|_p = y, p = q \cdot i, \mathrm{head}(x\theta|_q) = f, f : \mathsf{s}_1 \times \ldots \mathsf{s}_n \to \mathsf{s}\} \cup \{\mathsf{s} \mid x \in \mathrm{dom}(\theta), x\theta = y, x \text{ of sort } \mathsf{s}\}$.

Let σ be an arbitrary sort-preserving unifier of u and v. Thus $\sigma = \theta\sigma'$ for some substitution σ' with $\mathrm{dom}(\sigma') \subseteq \mathrm{var}(\theta)$. Consider an arbitrary variable $y' \in \mathrm{var}(\theta)$ and let s' be the sort of $y'\sigma'$. We prove next that $\mathsf{s}' \leq \mathsf{s}$, for any $\mathsf{s} \in S_{y'}$.

We distinguish two cases depending on how s was obtained. If s is the sort of some $x \in \mathrm{dom}(\theta)$ with $x\theta = y'$, then $x\sigma = y'\sigma'$ and hence $\mathsf{s}' \leq \mathsf{s}$, using the fact that σ is sort-preserving. If s is given by the sort of the function symbol f above y' in $x\theta$ for some $x \in \mathrm{dom}(\theta)$, then again $\mathsf{s}' \leq \mathsf{s}$, as $y'\sigma'$ is a subterm right below f in the well-sorted term $x\sigma$. Hence, in both cases, $\mathsf{s}' \leq \mathsf{s}$.

We define now ρ as the renaming which substitutes each variable y in $\mathrm{var}(\theta)$ with a new variable z of sort s_y, where s_y is least sort in S_y. Note that s_y is well defined. Indeed, suppose that s_1 and s_2 are two distinct minimums of the above set. Thus s_1 and s_2 are not comparable. Let s' be the sort of $y\sigma'$. We have shown that $\mathsf{s}' \leq \mathsf{s}_1$ and $\mathsf{s}' \leq \mathsf{s}_2$. This is in contradiction with the tree structure of Sorts.

Let $\theta_0 = \theta\rho$. For any $x \in \mathrm{dom}(\theta_0)$, $x\theta_0$ is well-sorted (this is given by the construction of ρ). And since the sort of $x\theta_0$ equals the sort of $x\sigma$, it follows that θ_0 is sort-preserving. Finally, $\sigma = \theta_0(\rho^{-1}\sigma')$ and $\rho^{-1}\sigma'$ is sort-preserving. Indeed, for any $z \in \mathrm{dom}(\rho^{-1}\sigma')$, $z\rho^{-1}\sigma'$ is well-sorted, being a subterm of $\mathrm{ran}(\sigma)$. And, as $z\rho^{-1}\sigma' = y\sigma'$ where $y = \rho^{-1}(z)$, the sort of $z\rho^{-1}\sigma'$ is smaller or equal than the sort of z (which is s_y).

As σ was arbitrarily chosen, it follows that θ_0 is a most general sort-preserving unifier of u and v. ∎

Exemple 1.4 *Note that if the basic sorts only form a lattice, the above proposition does not hold. Indeed, let us consider the 6 different sorts $\mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_1', \mathsf{s}_2, \mathsf{s}_2', \mathsf{s}_3$ with $\mathsf{s}_0 \geq \mathsf{s}_1$, $\mathsf{s}_1 \geq \mathsf{s}_1'$, $\mathsf{s}_2 \geq \mathsf{s}_2'$, $\mathsf{s}_1 \geq \mathsf{s}_2'$, $\mathsf{s}_2 \geq \mathsf{s}_1'$, $\mathsf{s}_1' \geq \mathsf{s}_3$, and $\mathsf{s}_2' \geq \mathsf{s}_3$. Taking $x : \mathsf{s}_1$ and $y : \mathsf{s}_2$, σ_1 and σ_2 given by $x\sigma_1 = y\sigma = z_1 : \mathsf{s}_1'$ and $x\sigma_2 = y\sigma_2 = z_2 : \mathsf{s}_2'$ are sort-preserving unfiers of x and y. However, there is no most-general sort-preserving unifier of x and y: suppose that there is such a unifier θ with $x\theta$ having some sort s. Then $\mathsf{s} \leq \mathsf{s}_1$, $\mathsf{s} \leq \mathsf{s}_2$ (as θ is sort-preserving). Also, as $z_1 : \mathsf{s}_1 = x\sigma_1 = (x\theta : \mathsf{s})\sigma'$ for some sort-preserving σ', it follows that $\mathsf{s}_1 \leq \mathsf{s}$. Hence $\mathsf{s} = \mathsf{s}_1$, and analogously $s = \mathsf{s}_2$, which constradicts the fact that the sorts s_1 and s_2 are different.*

A *context* with n holes is a function $\lambda x_1, \ldots, x_n.t$, where t is a term. The application of the context $C = \lambda x_1, \ldots, x_n.t$ to the terms $t_1, \ldots, t_n$ is $C[t_1, \ldots, t_n] \stackrel{\mathrm{def}}{=} t[{}^{t_1}\!/\!_{x_1}] \ldots [{}^{t_n}\!/\!_{x_n}]$ (where bounded variables $x_1, \ldots, x_n$ are first renamed if they also occur in t_i). When the variables x_i occur exactly once in t, they may be replaced by *holes*, denoted $[\,]$. In this case, we suppose that the variables are renamed such that the occurrence of x_i corresponds to the i-th hole (the order is given by a traversal of the tree-representation of t which respects the order of children). In other words, $C[t_1, \ldots, t_n]$ is obtained by replacing the i-th hole with t_i. The *empty context* is $\lambda x.x$ and is denoted by $[\,]$. We extend all definitions regarding terms to contexts $C = \lambda x_1, \ldots, x_n.t$ by applying them to t, while considering $\mathrm{var}(C) = \mathrm{var}(t) \setminus \{x_1, \dots, x_n\}$. For example, C is ground if $\mathrm{var}(t) = \{x_1, \dots, x_n\}$.

1.1.4 Equational theories and rewriting systems

An *equation* over the signature $\mathcal{F}$ is a unordered pair of terms $\{u, v\}$, denoted $u \doteq v$ (or $v \doteq u$), with $u, v \in \mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$ having the same sort. An *equational presentation* $\mathcal{E}$ over the signature $\mathcal{F}$ is a set of equations over $\mathcal{F}$, such that for any $(u \doteq v) \in \mathcal{E}$, $\mathrm{names}(u, v) = \emptyset$. The *equational theory* induced by $\mathcal{E}$ on $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$, denoted $=_{\mathcal{E}}$, is the smallest congruence such that $u\sigma =_{\mathcal{E}} v\sigma$ for all equations $(u \doteq v) \in \mathcal{E}$ and all substitutions σ. We often don't distinguish between an equational theory and its equational presentation.

A *rewrite rule* over the signature $\mathcal{F}$ is a pair of terms (u, v), denoted $u \to v$, with $u, v \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ having the same sort. A *rewriting system* $\mathcal{R}$ over the signature $\mathcal{F}$ is a set of rewrite rules over $\mathcal{F}$.

A term t *rewrites to* or *reduces to* a term t', denoted by $t \to t'$, if there exists a position $p \in \mathrm{pos}(t)$, a rewrite rule $(u \to v) \in \mathcal{R}$ and a substitution σ such that $t|_p = u\sigma$ and $t' = t[v\sigma]_p$. We may write $t \xrightarrow{p}_{\mathcal{R}} t'$ to specify the position p at which the reduction takes place. We may also drop the subscript and simply write $t \to t'$ if $\mathcal{R}$ is clear from the context. The induced relation on terms $\to_{\mathcal{R}}$ is called the reduction relation, $t \to_{\mathcal{R}} t'$ is also called a reduction step, and a sequence of reduction steps is called a reduction sequence. The relation $\leftrightarrow_{\mathcal{R}}$ is the symmetric closure of $\to_{\mathcal{R}}$.

The equational theory associated to a rewriting system $\mathcal{R}$ is $\mathcal{E}(\mathcal{R}) \stackrel{\text{def}}{=} \{u \doteq v \mid (u \to v) \in \mathcal{R}\}$. By Birkhoff's theorem (1935), we have $u \stackrel{*}{\leftrightarrow}_{\mathcal{R}} v$ if and only if $u =_{\mathcal{E}(\mathcal{R})} v$. The other way around, the rewriting system associated with an equational theory $\mathcal{E}$ is

$$\mathcal{R}(\mathcal{E}) \stackrel{\text{def}}{=} \{u \to v \mid (u \doteq v) \in \mathcal{E} \vee (v \doteq u) \in \mathcal{E}\}.$$

That is, $\mathcal{R}(\mathcal{E})$ is obtained by orienting in both directions the equations of $\mathcal{E}$. We then have $u =_{\mathcal{E}} v$ if and only if $u \stackrel{*}{\leftrightarrow}_{\mathcal{R}(\mathcal{E})} v$.

A term t is a or in *normal form* iff there is no term t' such that $t \to_{\mathcal{R}} t'$. We say that a term t has a normal form iff there is a normal form t' such that $t \to_{\mathcal{R}}^* t'$. A normal form of t is denoted $t{\downarrow}$.

A rewriting system is *terminating* iff any reduction sequence is finite. A rewriting system is *confluent* if for any terms t, u, v such that $t \to_{\mathcal{R}}^* u$ and $t \to_{\mathcal{R}}^* v$ there exists a term w such that $u \to_{\mathcal{R}}^* w$ and $v \to_{\mathcal{R}}^* w$. In a confluent rewriting system, if a term has a normal form then this is unique. A rewriting system is *convergent* iff it is confluent and terminating.

Remark We will sometimes need (e.g. in Chapter 4) to extend the notion of substitution of variables to substitution of names. We do it in the expected way. However, without explicit mention "substitution" will only refer to variable substitutions and substitutions of names will only be explicit (as in $[{}^{t}/_{n}]$).

Nevertheless, we need to make sure that equational theories remain stable by substitution. Formally, an equational theory $\mathcal{E}$ is *stable by substitution of names* if for any terms u, v, t and name n, $u =_{\mathcal{E}} v$ implies $u[{}^{t}/_{n}] =_{\mathcal{E}} v[{}^{t}/_{n}]$.

Proposition 1.5 *Any equational theory is stable by substitution of names.*

Proof Consider an arbitrary equational theory and its equational presentation $\mathcal{E}$. Let u, v, and t be arbitrary terms such that $u =_{\mathcal{E}} v$, and n be a name. Since we have that $u \stackrel{*}{\leftrightarrow}_{\mathcal{R}(\mathcal{E})} v$, it is sufficient to prove that $u \to_{\mathcal{R}(\mathcal{E})} v$ implies $u[{}^{t}\!/_{n}] \to_{\mathcal{R}(\mathcal{E})} v[{}^{t}\!/_{n}]$. Indeed, a simple induction on the length of the reduction sequence would then finish the proof.

Hence, suppose that $u \to_{\mathcal{R}(\mathcal{E})} v$. There is then a rule $(l \to r) \in \mathcal{R}(\mathcal{E})$ such that $u|_p = l\sigma$ and $v = u[r\sigma]_p$ for some position p in u, and some substitution σ. We have $u[{}^{t}\!/_{n}]|_p = (u|_p)[{}^{t}\!/_{n}] = (l\sigma)[{}^{t}\!/_{n}] = l(\sigma[{}^{t}\!/_{n}])$. The non-trivial equality here is the last one, which holds since $\text{names}(l) = \emptyset$ (by the definition of equational presentations). Similarly, we obtain that $v[{}^{t}\!/_{n}] = u[r(\sigma[{}^{t}\!/_{n}])]_p$. Thus, $u[{}^{t}\!/_{n}] \to_{\mathcal{R}(\mathcal{E})} v[{}^{t}\!/_{n}]$. ∎

1.1.5 Term deduction systems

A *deduction expression* over the signature $\mathcal{F}$ is a pair (S, u) denoted $S \vdash u$, where S is a special variable (not in $\mathcal{X}$) and $u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is a term. A *deduction rule* over the signature $\mathcal{F}$ is a triple (w, e, C), denoted

$$\frac{e_1 \quad \dots \quad e_k}{e}\; C$$

where $w = e_1 \dots e_k$, with $k \geq 0$, is a word over deduction expressions over $\mathcal{F}$, e is a deduction expression over $\mathcal{F}$, and C is a predicate. The expressions e_i, with $1 \leq i \leq k$, are called *premisses*, e is called *conclusion*, and C is called *condition* of a deduction rule. A rule with no premisses is called an *axiom*. We omit the condition C when C is the **true** predicate (that always holds), and we say in this case that the rule is *without condition*. A *deduction system* over the signature $\mathcal{F}$ is a set of deduction rules over $\mathcal{F}$. A deduction system is without conditions if each of its rules is without condition or is the axiom $\frac{}{S \vdash x}\; x \in S$.

For a set of terms T and a term u, a *proof* of $T \vdash u$ in a deduction system $\mathcal{I}$ is a tree having nodes labeled by expressions $T \vdash v$, where v are terms, such that the root is labeled by $T \vdash u$ and if the node $T \vdash v$ has $k \geq 0$ children $T \vdash v_1, \dots, T \vdash v_k$ then there is a deduction rule

$$\frac{S \vdash t_1 \quad \dots \quad S \vdash t_k}{S \vdash t}\; C$$

in $\mathcal{I}$ and a substitution σ such that $t\sigma = v$, $t_1\sigma = v_1, \dots, t_k\sigma = v_k$ and $C[{}^{T}\!/_{S}]\sigma$ holds. The *size* of a proof is given by the number of its nodes. A *minimal* proof of $T \vdash u$ is a proof with a minimal number of nodes (among all the proofs of $T \vdash u$).

We sometimes think of a proof as the line graph[4] of the tree defined above. In this case the meaning of nodes and edges would be reversed (that is, nodes would be instances of rules, and edges would be deduction expressions). It will be clear from the context which definition is used.

We say that u is *deducible* from T in $\mathcal{I}$, and write $T \vdash_{\mathcal{I}} u$, if there is a proof of $T \vdash u$ in $\mathcal{I}$. We may omit mentioning the deduction system $\mathcal{I}$ if it is clear from the context.

1.2 Cryptographic primitives and messages

We start this section by presenting some intuitions behind the use of two related but slightly different representations of some cryptographic primitives. We then present a sort system and a signature that are very common in cryptographic protocols and which are largely used in this thesis. Finally, we define two related deduction systems (both used) and a redundant one (not used), and we conclude by discussing the corresponding deduction problem.

[4] If $G = (V, E)$ is a graph then the *line graph* of G is $L(G) \stackrel{\text{def}}{=} (E, \{\{e, f\} \mid e \text{ and } f \text{ are adjacent in } G\})$.

Some intuitions Cryptographic primitives can be seen as functions from messages to messages[5] and messages themselves are obtained by applying operations to other messages. It is hence natural to abstract cryptographic primitives by function symbols and messages by terms. There is also the need to specify the process of applying these operations. This is done abstractly by deduction rules which define how terms can be obtained from other terms. As an example, consider the encryption of a message (named) m by a key (named) k. The encryption primitive is represented by the function symbol enc and the encrypted message by the term $\mathsf{enc}(m,k)$ where m and k are here terms. The process of obtaining this term is described by (an instance of) the rule[6]

$$\frac{x \qquad y}{\mathsf{enc}(x,y)}$$

Some primitives build "new" messages, in the sense that the obtained message cannot be expressed by a term without using a dedicated function symbol (like enc for the encryption). These function symbols are hence called *constructors*. For the other primitives, thus called *destructors*, on the contrary, one need not necessarily use a dedicated symbol (at least in some situations, as we explain below). These primitives are typically "inverse" operations, for which by applying in sequence the operation and its inverse on some message m one obtains (under some conditions) m, or some parts of it.

For example, by decrypting an encrypted message with the right key one obtains the plain message. If these conditions are not satisfied (for example, when one tries to decrypt using the wrong key), or if the inverse operation is applied on a arbitrary message, the results are unexpected. More exactly, they are implementation dependent. There are mainly two possibilities.

One is to consider that these inverse operations cannot be applied at all, or that they lead to "junk" that is immediately recognisable. In this case, only the correct use of the inverse operation is modeled. Thus, it is not necessarily to represent these inverse operations by new function symbols, since their application always leads to existing messages. For example, the application of decryption is represented implicitly by

$$\frac{\mathsf{enc}(x,y) \qquad y}{x}$$

The other possibility is to consider that the application of inverse operations is always feasible and it leads to "valid" messages, even if they are not "intelligible" (for example, $\mathsf{dec}(a,k)$ the decryption of some identity a with the key k is to everybody just an (almost) random sequence of bits). In this case, the inverse operations are modeled by function symbols and their application is explicitly modeled by deduction rules. For example, applying decryption with the right key to an encrypted message leads to the following derivation:

$$\frac{\mathsf{enc}(x,y) \qquad y}{\mathsf{dec}(\mathsf{enc}(x,y),y)}$$

But, when the conditions are satisfied, the message represented by the new term (obtained as the result of the application of the deduction rule) is some existing message. Indeed, in our example, the message m is now represented by the two terms $\mathsf{dec}(\mathsf{enc}(m,k),k)$ and m. We need hence to equate the new term with the term already representing the initial message. This is done by

[5] Seen at low level, cryptographic primitives are algorithms having bitstrings as inputs and outputs.

[6] For the sake of the presentation the "deduction rules" on this page are deliberately simplified (see the definition in Section 1.1.5).

the means of rewrite rules. For example the rewrite rule $\mathsf{dec}(\mathsf{enc}(x, y), y) \rightarrow x$ says that after encrypting and then decrypting a message with the same key the same message is obtained.

Recapitulating, the explicit presence of destructors in the model depends on whether we consider that the corresponding operations may clearly fail (or produce detectable "junk"), or that they do not fail and produce "valid" messages (or non-detectable "junk").

Besides the properties representing their basic functionality, cryptographic primitives may exhibit other properties. Among them, *algebraic properties*, like associativity, commutativity, nilpotence, etc., are always expressed by equations. They induce equivalence classes on the set of terms, and indeed, due to these properties the same message is represented by syntactically different terms. For example, the terms $\langle a, \langle b, c\rangle\rangle$ and $\langle\langle a, b\rangle, c\rangle$ model the same message, the concatenation of a, b, and c. Another example we have already seen in the Introduction is encryption in ECB mode, which is homomorphic (w.r.t. the concatenation). This is expressed by the equation $\{\!\{\langle x, y\rangle\}\!\}_z \doteq \langle\{\!\{x\}\!\}_z, \{\!\{y\}\!\}_z\rangle$.

A summary of the above discussion is given by the Table 1.1.

real world	symbolic world
messages	terms
cryptographic primitives	function symbols
operations that never fail	constructors, explicit destructors
operations that may fail	implicit destructors
basic functionalities of crypto. prim.	rewrite rules
algebraic properties of crypto. prim.	equations
applying cryptographic primitives	deduction rules

Table 1.1: Analogy between the real world and the symbolic world.

1.2.1 A sort system for cryptographic protocols

In some models, it may be the case that there are terms that do not represent "valid" messages (e.g. $dec(a, k)$). Also, it is usually supposed that an identity cannot be confused with other data, like nonces or ciphertexts. A simple way to eliminate such undesired situations, is to fix a sort system and associate a sort to each variable, name and function symbol.

We consider signatures with the following sorts: a sort Id for agent identities, and sorts Int, Nonce, Rand and Time for integers, nonces, randomness used in probabilistic encryption, and timestamps respectively. These four sorts could be in fact represented by a single sort, for example Int, the values that terms of these sorts represent are in fact all integers. But for the sake of clarity and flexibility we use this presentation. For keys we consider the sorts $\mathsf{SigKey}, \mathsf{VerKey}, \mathsf{EncKey}, \mathsf{DecKey}, \mathsf{SymKey}$ representing keys used in signing, signature verification, public-key encryption, public-key decryption, and symmetric encryption algorithms respectively. We also use sorts $\mathsf{Ciphertext}$, $\mathsf{Signature}$, and Pair for ciphertexts, signatures, and pairs, respectively. The sort Msg is a supersort containing all other sorts enumerated above.

In summary, throughout this thesis we consider the following basic sorts

$$\begin{aligned}\mathsf{Sorts}_0 = \{&\mathsf{Id},\ \mathsf{Int}, \mathsf{Nonce}, \mathsf{Rand}, \mathsf{Time},\ \mathsf{SymKey}, \mathsf{EncKey}, \mathsf{DecKey}, \mathsf{SigKey}, \mathsf{VerKey},\\ &\qquad\qquad \mathsf{PubKey}, \mathsf{PrivKey},\ \mathsf{Ciphertext}, \mathsf{Signature}, \mathsf{Pair}, \mathsf{Msg}\},\end{aligned}$$

with $\mathsf{s} \leq \mathsf{Msg}$ for all $\mathsf{s} \in \mathsf{Sorts}$, and $\mathsf{EncKey}, \mathsf{VerKey} \leq \mathsf{PubKey}$, $\mathsf{DecKey}, \mathsf{SigKey} \leq \mathsf{PrivKey}$. However, we do not require that sorts are different. Indeed, in some situations we consider only one sort (e.g. in Chapters 3 and 4), that is $\mathsf{s} = \mathsf{Msg}$ for all $\mathsf{s} \in \mathsf{Sorts}$, while in others sorts are different.

1.2.2 A signature for cryptographic protocols

Table 1.2 lists most of the function symbols used in this thesis.

function symbol	sort	description
k	$\mathsf{Id} \times \mathsf{Id} \to \mathsf{SymKey}$	symmetric key
ek	$\mathsf{Id} \to \mathsf{EncKey}$	asym. encryption key
dk	$\mathsf{Id} \to \mathsf{DecKey}$	asym. decryption key
sk	$\mathsf{Id} \to \mathsf{SigKey}$	signing key
vk	$\mathsf{Id} \to \mathsf{VerKey}$	verification key
pub	$\mathsf{Id} \to \mathsf{PubKey}$	public key
priv	$\mathsf{Id} \to \mathsf{PrivKey}$	private key
encd (or $\{\!\{\_\}\!\}\_$)	$\mathsf{Msg} \times \mathsf{SymKey} \to \mathsf{Ciphertext}$	det. sym. encryption
enc	$\mathsf{Msg} \times \mathsf{SymKey} \times \mathsf{Rand} \to \mathsf{Ciphertext}$	prob. sym. encryption
dec	$\mathsf{Ciphertext} \times \mathsf{SymKey} \to \mathsf{Msg}$	sym. decryption
encad (or $\{[\_]\}\_$)	$\mathsf{Msg} \times \mathsf{EncKey} \to \mathsf{Ciphertext}$	det. asym. encryption
enca	$\mathsf{Msg} \times \mathsf{EncKey} \times \mathsf{Rand} \to \mathsf{Ciphertext}$	prob. asym. encryption
deca	$\mathsf{Ciphertext} \times \mathsf{DecKey} \to \mathsf{Msg}$	asym. decryption
sign (or $[\![\_]\!]\_$)	$\mathsf{Msg} \times \mathsf{SigKey} \to \mathsf{Signature}$	signature
check	$\mathsf{Msg} \times \mathsf{Signature} \times \mathsf{VerKey} \to \mathsf{Msg}$	check signature
$\mathsf{retrieve}$	$\mathsf{Signature} \to \mathsf{Msg}$	retrieve signed message
h	$\mathsf{Msg} \to \mathsf{Msg}$	hash
pair (or $\langle \_, \_ \rangle$)	$\mathsf{Msg} \times \mathsf{Msg} \to \mathsf{Pair}$	pair
fst (or π_1)	$\mathsf{Pair} \to \mathsf{Msg}$	1^{st} projection
snd (or π_2)	$\mathsf{Pair} \to \mathsf{Msg}$	2^{nd} projection
$\mathsf{ok}, \mathsf{init}, \mathsf{stop}, \mathsf{fake}$	Msg	special constants

Table 1.2: The set $\mathcal{F}_0$ of function symbols with their arities

The four operations $\mathsf{ek}, \mathsf{dk}, \mathsf{sk}, \mathsf{vk}$ are defined on the sort Id and return the asymmetric encryption key, asymmetric decryption key, signing key, and verification key associated to the input identity. The two function symbols pub and priv represent public and respectively private keys for both encryption and digital signatures. They are safe abstractions of ek and vk on the one side and of dk and sk on the other side.

We then have function symbols for symmetric and asymmetric encryption and decryption. We model both deterministic and probabilistic encryption. The latter one has a third parameter (besides the message to be encrypted and the encryption key) which represents the randomness used to obtain the nondeterminism: two encryptions of the same messages under the same keys are different (if they use different randomness).

Next, sign represents the operation of digital signing of a message, check that of verifying a signature and $\mathsf{retrieve}$ that of obtaining the signed message from the signature.

For deterministic encryption and digital signatures we usually use the classical notation with brackets (for example $\{\!|m|\!\}_k$ instead of $\mathsf{encd}(m,k)$). The convention is that the exterior brackets say whether the function symbol represents an encryption (and we use $\{\}$) or a digital signature (and we use $[\,]$), while the interior brackets say whether it is symmetric (use of $\{\}$) or asymmetric (use of $[\,]$) operation.

The function symbol h models the operations of hashing of a message.

The symbol pair represents the pairing function, while fst and snd are the associated projection functions. We abbreviate $\mathsf{pair}(x,y)$ by $\langle x,y\rangle$, $\mathsf{fst}(x)$ and $\mathsf{snd}(x)$ by $\pi_1(x)$ and $\pi_2(x)$ respectively. We also suppose that pairing is left-associative and hence write $\langle x,y,z\rangle$ for $\langle\langle x,y\rangle,z\rangle$. Moreover, we may even entirely omit the angle brackets when the pairing function symbol is not in head position. For example, we may write $\{\!|a,b|\!\}_k$ instead of $\{\!|\langle a,b\rangle|\!\}_k$.

Finally, the signature contains a few special constants which usually do not represent real messages but are useful for specifying protocols. For example, the constant ok represents a special message issued as a result of some successful testing operation.

Operations on messages can fall into two classes: *public* and *private* operations, depending on whether they can be performed by any party, or only by some parties. We obtain accordingly a partition of function symbols into public and private function symbols, denoted by

$$\mathcal{F} = \mathcal{F}_{\mathsf{pub}} \uplus \mathcal{F}_{\mathsf{priv}}.$$

We assume that any signature is partitioned in this way. We suppose that for each sort there is an infinite number of public constants and an infinite number of private constants of that sort. A term t is *public* if $t \in \mathcal{T}(\mathcal{F}_{\mathsf{pub}},\mathcal{X},\mathcal{N})$.

In our context, the only private operations are those of obtaining the private keys associated to identities. Indeed, we have $\{\mathsf{k},\mathsf{dk},\mathsf{sk},\mathsf{priv}\} \subseteq \mathcal{F}_{\mathsf{priv}}$ and all other function symbols in $\mathcal{F}_0$ are public symbols (that is, they are in $\mathcal{F}_{\mathsf{pub}}$). To model asymmetric keys, we could have eliminated private symbols by using names like sk_a instead of $\mathsf{priv}(a)$ (and $\mathsf{pub}(sk_a)$ instead of $\mathsf{pub}(a)$). However, we prefer to use names only for fresh data.

Throughout this thesis we deal with four (one for each of the following chapters) slightly different order-sorted signatures. These will be obtained by fixing a sort system (basically by equating some sorts in Sorts_0*) and a set of function symbols (mainly by considering a subset of* $\mathcal{F}_0$ *together with a set of constants for agent identities and nonces).*

1.2.3 Two deduction systems for cryptographic protocols

As we have already mentioned, the application of operations on messages and hence the construction of new messages are modeled by deduction rules. And, depending on whether operations which are supposed to work only in certain conditions (e.g. when using the right key) can only be applied when "successful", or can always be applied, we have two types of deduction systems. However the functionality of operations is the same in the two cases, and we model it by a *simple* rewriting system.

Definition 1.6 (Constructors, destructors, simple rewriting systems)
Let $\mathcal{F} = \mathcal{F}_{\mathsf{pub}} \uplus \mathcal{F}_{\mathsf{priv}}$ *be a signature and let* $\mathcal{R}$ *be a rewriting system over* $\mathcal{T}(\mathcal{F},\mathcal{X})$ *with* $\mathrm{var}(r) \subseteq \mathrm{var}(l)$ *and* $l \notin \mathcal{X}$ *for all* $(l \to r) \in \mathcal{R}$.

The head symbols of left hand sides of rewrite rules in $\mathcal{R}$ *are called* destructors*. All other function symbols in* $\mathcal{F}_{\mathsf{pub}}$ *are called* constructors*. This partition is denoted* $\mathcal{F}_{\mathsf{pub}} = \mathcal{F}_{\mathsf{cstr}} \uplus \mathcal{F}_{\mathsf{dstr}}$.

We say that $\mathcal{R}$ *is* simple *if every rule is of the form* $g(l_1,\ldots,l_n) \to r$ *with* $g \in \mathcal{F}_{\mathsf{pub}}$, $n \geq 0$, $\{r,l_1,\ldots,l_n\} \subseteq \mathcal{T}(\mathcal{F}\setminus\mathcal{F}_{\mathsf{dstr}},\mathcal{X})$*, and* g *does not occur in the other rules of* $\mathcal{R}$.

The rewriting system with regard to which constructors and destructors are defined will often be implicit.

Proposition 1.7 *A simple rewriting system is convergent.*

Proof Termination follows from the fact that at each reduction step a destructor is eliminated and terms without destructors are in normal form. Confluence follows immediately because the head function symbol of a rule does not occur elsewhere in the rewriting system, and thus there are no critical pairs (see [BN98] for a definition of critical pairs and their use for testing (local) confluence). ∎

Intuitively, $T \vdash u$ means that an agent is able to compute the message u from the set of messages T. Any deduction system allows one to use known messages and to compose known messages.

Let $F \subseteq \mathcal{F}$ be a set of function symbols. A *minimal deduction system* w.r.t. F, denoted $\mathcal{I}_F$ is given by the following rules:

$$(\in)\ \frac{}{S \vdash x}\ x \in S \qquad\qquad (\mathsf{Comp}_f)\ \frac{S \vdash x_1 \quad \ldots \quad S \vdash x_n}{S \vdash f(x_1, \ldots, x_n)}\ f \in F$$

The labels given between parentheses in the left hand side of rules are just names used for refering to that rule. The above rules are also called *membership* and *composition* rules, respectively. Remark that the latter is in fact a rule schema, i.e. there is a corresponding composition rule (without condition) for each $f \in F$.

Usually, F is either $\mathcal{F}_{\mathsf{cstr}}$ or $\mathcal{F}_{\mathsf{pub}}$. We may omit F in $\mathcal{I}_F$ when it is clear from the context. Also note that from the point of view of building terms all function symbols in F are "constructors". However, for $F = \mathcal{F}_{\mathsf{cstr}}$, only constructors (in the sense of Definition 1.6) are used for building terms.

Given a simple rewriting system $\mathcal{R}$, we associate to each rewrite rule $(l \to r) \in \mathcal{R}$ with $l = g(l_1, \ldots, l_n)$ the following deduction rule:

$$(\mathsf{rw}_{l \to r})\ \frac{S \vdash l_1 \quad \ldots \quad S \vdash l_n}{S \vdash r}$$

Definition 1.8 ($\mathcal{I}(\mathcal{R})$) *Let $\mathcal{R}$ be a simple rewriting system over the signature $\mathcal{F}$. We define by*

$$\mathcal{I}(\mathcal{R}) \stackrel{\text{def}}{=} \mathcal{I}_{\mathcal{F}_{\mathsf{cstr}}} \cup \{(\mathsf{rw}_{l \to r}) \mid (l \to r) \in \mathcal{R}\}$$

the deduction system associated with $\mathcal{R}$.

To an equational theory $\mathcal{E}$ we associate the following deduction rule:

$$(\mathsf{Eq}_{\mathcal{E}})\ \frac{S \vdash x}{S \vdash y}\ x =_{\mathcal{E}} y$$

We may drop the subscript when $\mathcal{E}$ is clear, and write (Eq) instead.

Definition 1.9 ($\mathcal{I}(\mathcal{E})$) *Let $\mathcal{E}$ be an equational theory over the signature $\mathcal{F}$. We define by*

$$\mathcal{I}(\mathcal{E}) \stackrel{\text{def}}{=} \mathcal{I}_{\mathcal{F}_{\mathsf{pub}}} \cup \{(\mathsf{Eq}_{\mathcal{E}})\}$$

the deduction system associated with $\mathcal{E}$.

The following known lemma (see e.g. [AC04]) characterises $\mathcal{I}(\mathcal{E})$ in terms of public contexts.

Lemma 1.10 *Consider an equational theory $\mathcal{E}$. Let $T = \{t_1, \ldots, t_n\}$ be finite set of terms and u, v be terms. Then $T \vdash_{\mathcal{I}(\mathcal{E})} u$ if and only if there exists a public context C such that $C[t_1, \ldots, t_n] =_{\mathcal{E}} u$.*

The deduction systems $\mathcal{I}(\mathcal{R})$ and $\mathcal{I}(\mathcal{E})$ represent two ways of reasoning about the applications of cryptographic primitives to messages. In the former, deduction system destructors operations are implicit and are matched against the right patterns, while in the latter, destructor operations are explicit and terms are equated when possible. We sometimes call them deduction system "with matching", and respectively deduction system "with equalities" or with "explicit destructors". Note that $\mathcal{I}(\mathcal{R})$ is a deduction system over $\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}$. We show next that considering deductions only over $\mathcal{T}(\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}, \mathcal{X}, \mathcal{N})$ the two deductions systems are "equivalent".

Let $\mathcal{T}$ be a set of terms. We say that two deduction systems $\mathcal{I}$ and $\mathcal{I}'$ are *equivalent* over $\mathcal{T}$ if for all sets of terms T and terms t, with $T \cup \{t\} \subseteq \mathcal{T}$, we have $T \vdash_{\mathcal{I}} t$ if and only if $T \vdash_{\mathcal{I}'} t$.

Proposition 1.11 *Let $\mathcal{F}$ be a signature and $\mathcal{R}$ a simple rewriting system. The deduction systems $\mathcal{I}(\mathcal{R})$ and $\mathcal{I}(\mathcal{E}(\mathcal{R}))$ are equivalent over $\mathcal{T}(\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}, \mathcal{X}, \mathcal{N})$.*

Proposition 1.11 assumes a simple rewriting system $\mathcal{R}$ and shows that the two deduction systems derived from $\mathcal{R}$ are equivalent. S. Delaune [Del06] proved a similar statement, but starting from an arbitrary deduction system $\mathcal{I}$ without conditions and building an equivalent deduction system $\mathcal{I}(\mathcal{E}(\mathcal{R}_{\mathcal{I}}))$ where $\mathcal{R}_{\mathcal{I}}$ is a simple rewriting system. Indeed for each deduction rule $\frac{S \vdash l_1 \quad \ldots \quad S \vdash l_n}{S \vdash r}$ in $\mathcal{I}$ one can add the rewrite rule $g(l_1, \ldots, l_n) \doteq r$ in $\mathcal{R}_{\mathcal{I}}$, where g is a new function symbol. However, the proofs are the same, since S. Delaune's transformation (from $\mathcal{I}$ to $\mathcal{R}_{\mathcal{I}}$) and this transformation (from $\mathcal{R}$ to $\mathcal{I}(\mathcal{R})$) are inverse to one another. Rephrasing the mentioned result in our formalism, we obtain the following proposition.

Proposition 1.12 ([Del06]) *Let $\mathcal{F}$ be a signature. For every deduction system $\mathcal{I}$ over $\mathcal{F}$ without conditions, there exist a set of public function symbols $\mathcal{G}$ with $\mathcal{G} \cap \mathcal{F} = \emptyset$, and a simple rewriting system $\mathcal{R}$ over $\mathcal{F} \cup \mathcal{G}$ such that $\mathcal{I}$ and $\mathcal{I}(\mathcal{E}(\mathcal{R}))$ are equivalent over $\mathcal{T}(\mathcal{F}, \mathcal{X}, \mathcal{N})$.*

Algebraic properties of primitives are not considered in this thesis. We just mention that they are usually represented by a set of equations $\mathcal{E}_{alg}$. In the same spirit, one considers one of the following two deduction systems: $\mathcal{I}(\mathcal{R}) \cup \{(\mathsf{Eq}_{\mathcal{E}_{alg}})\}$ or $\mathcal{I}(\mathcal{E}(\mathcal{R}) \cup \mathcal{E}_{alg})$.

Note that when working on $\mathcal{T}(\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}, \mathcal{X}, \mathcal{N})$ we may simply consider that the signature is $\mathcal{F}_{\mathsf{cstr}} \cup \mathcal{F}_{\mathsf{priv}}$, and work directly with $\mathcal{I}(\mathcal{R})$ (without mentioning the rewriting system $\mathcal{R}$ and (explicit) destructors). We will do this in Chapters 2, 3 and 5.

1.2.3.1 Example. The Dolev-Yao rules

The rewrite rules in Figure 1.2 constitute an example of simple rewriting system, denoted $\mathcal{R}_0$. By Proposition 1.7, $\mathcal{R}_0$ is convergent and so is any of its subsets.

The deduction rules of Figure 1.3, together with the membership and composition rules represent the deduction system with matching associated with $\mathcal{R}_0$, denoted $\mathcal{I}_0 \stackrel{\text{def}}{=} \mathcal{I}(\mathcal{R}_0)$.

The first two deduction rules (together with the corresponding composition rules) are known as the *standard Dolev-Yao rules.* By extension, the rules of a deduction system with matching are also known as Dolev-Yao rules. The rules in Figure 1.3 are also known as *decomposition*

$\mathsf{fst}(\mathsf{pair}(x,y)) \to x$	1^{st} projection
$\mathsf{snd}(\mathsf{pair}(x,y)) \to y$	2^{nd} projection
$\mathsf{dec}(\mathsf{encd}(x,y),y) \to x$	sym. decryption
$\mathsf{deca}(\mathsf{encad}(x,\mathsf{ek}(y)),\mathsf{dk}(y)) \to x$	asym. decryption
$\mathsf{retrieve}(\mathsf{sign}(x,y)) \to x$	retrieving the signed message

Figure 1.2: The rewriting system $\mathcal{R}_0$

$$(\mathsf{Proj}_i)\ \frac{S \vdash \mathsf{pair}(x_1,x_2)}{S \vdash x_i}\ i \in \{1,2\} \qquad 1^{\text{st}} \text{ and } 2^{\text{nd}} \text{ projections}$$

$$(\mathsf{Dec})\ \frac{S \vdash \mathsf{encd}(x,y) \qquad S \vdash y}{S \vdash x} \qquad \text{sym. decryption}$$

$$(\mathsf{Deca})\ \frac{S \vdash \mathsf{encad}(x,\mathsf{ek}(y)) \qquad S \vdash \mathsf{dk}(y)}{S \vdash x} \qquad \text{asym. decryption}$$

$$(\mathsf{Retr})\ \frac{S \vdash \mathsf{sign}(x,\mathsf{sk}(y))}{S \vdash x} \qquad \text{retrieving the signed message}$$

Figure 1.3: The deduction system $\mathcal{I}_d$

rules. Intuitively, these rules say that an agent can decompose messages by projecting, or by decrypting provided it has the decryption keys.

Concerning digital signatures, an agent is also able to verify whether a signature $\mathsf{sign}(m,k)$ and a message m match (provided he has the verification key). This operation is represented by a rewrite rule of the form

$$\mathsf{check}(x,\mathsf{sign}(x,\mathsf{sk}(y)),\mathsf{vk}(y)) \to \mathsf{ok}$$

Note that checking a signature does not lead to the generation of new message (ok is a public constant). This is why the corresponding deduction rule

$$(\mathsf{Chk})\ \frac{S \vdash x \qquad S \vdash \mathsf{sign}(x,\mathsf{sk}(y)) \qquad S \vdash \mathsf{vk}(y)}{S \vdash \mathsf{ok}}$$

is usually not present in deduction systems which model the intruder's capabilities, and we will also omit it (i.e. the rule (Chk)).

The rule (Retr) expresses that an agent can retrieve the whole message from its signature. This property may or may not hold depending on the signature scheme, and hence this rule is usually optional. Note that this rule is necessary for obtaining soundness properties w.r.t. cryptographic digital signatures [CW05].

Exemple 1.13 *The term $\langle k_1,k_2\rangle$ is deducible from the set $T = \{\{\!|k_1|\!\}_{k_2}, k_2\}$ in $\mathcal{I}_0$. A proof of $T \vdash_{\mathcal{I}_0} \langle k_1,k_2\rangle$ is:*

$$\frac{\dfrac{T \vdash \{\!|k_1|\!\}_{k_2} \quad T_1 \vdash k_2}{T \vdash k_1} \qquad T \vdash k_2}{T \vdash \langle k_1,k_2\rangle}$$

While the first formal models of security protocols in the symbolic world used equations (and hence explicit destructors) [DY83, EG83][7], most of the later literature on protocol verification did not use them. Instead, a deduction system with matching (or an equivalent model) was directly the starting point. Destructors have reappeared with the treatment of algebraic properties. Indeed, having a single set of equations modeling all properties (algebraic or not) of cryptographic primitives offers a more uniform approach (see for example the applied-pi calculus and its motivations [AF01]).

In each of the following chapters we will implicitly fix a certain rewriting system[8] $\mathcal{R}'$ and work explicitly with $\mathcal{I}(\mathcal{R}')$, or $\mathcal{I}(\mathcal{E}(\mathcal{R}'))$, or still another equivalent representation of deduction. More precisely, we use the deduction system with matching $\mathcal{I}_0$ in Chapters 2 and 5, a deduction system with equalities in Chapter 4, and Horn clauses in Chapter 3.

1.2.4 On the use of a third deduction system

In this section, we introduce still another deduction system, but show that it is redundant and hence we do not use it in the rest of the thesis.

For a rewriting system $\mathcal{R}$ we consider the following deduction rule:

$$(\mathsf{Rw}_{\mathcal{R}})\ \frac{S \vdash x}{S \vdash y}\ x \to^*_{\mathcal{R}} y$$

We define $\mathcal{I}^d(\mathcal{R}) \stackrel{\text{def}}{=} \mathcal{I}_{\mathcal{F}_{\mathsf{pub}}} \cup \{(\mathsf{Rw}_{\mathcal{R}})\}$.

The following lemma characterises $\mathcal{I}^d(\mathcal{R})$ in terms of public contexts, in the same manner as Lemma 1.10 does for $\mathcal{I}(\mathcal{E})$.

Lemma 1.14 *Let $\mathcal{R}$ be a rewriting system. Let $T = \{t_1, \ldots, t_n\}$ be finite set of terms and u, v be terms. Then $T \vdash_{\mathcal{I}^d(\mathcal{R})} u$ if and only if there exists a public context C such that $C[t_1, \ldots, t_n] \to^*_{\mathcal{R}} u$.*

Proof Suppose that $T \vdash_{\mathcal{I}^d(\mathcal{R})} u$. We reason by induction on the depth of a proof. Consider the rule applied in the root of the proof:

- If it is an axiom then take $C = x_i$ if the rule is $(\in)$ and $u = t_i$, and take $C = u$ if the rule is (Comp_f) with f of arity 0 (in this case u is a public constant).
- If it is the rule (Comp_f) with $f \in \mathcal{F}_{\mathsf{pub}}$ having arity $r \geq 1$ and the instances of the premisses being $T \vdash_{\mathcal{I}^d(\mathcal{R})} u_1, \ldots, T \vdash_{\mathcal{I}^d(\mathcal{R})} u_r$ then by induction hypothesis there exist public contexts $C_1, \ldots, C_r$ such that $C_i[t_1, \ldots, t_n] \to^*_{\mathcal{R}} u_i$, for all $1 \leq i \leq r$. We take $C = f(C_1, \ldots, C_r)$ and then $C[t_1, \ldots, t_n] \to^*_{\mathcal{R}} f(u_1, \ldots, u_r) = u$.
- If it is the rule $(\mathsf{Rw}_{\mathcal{R}})$ with the instance of the premise being $T \vdash_{\mathcal{I}^d(\mathcal{R})} u'$ then by induction hypothesis there exists a public context C' such that $C'[t_1, \ldots, t_n] \to^*_{\mathcal{R}} u'$. Since $u' \to^*_{\mathcal{R}} u$ we take $C = C'$.

For the converse direction, consider a public context C such that $C[t_1, \ldots, t_n] \to^*_{\mathcal{R}} u$. Since C is public and $t_i \in T$, there is a proof with the root labeled by $T \vdash C[t_1, \ldots, t_n]$ in $\mathcal{I}$. Adding the corresponding instance of the rule $(\mathsf{Rw}_{\mathcal{R}})$ to this tree we obtain a proof of $T \vdash u$ in $\mathcal{I}^d(\mathcal{R})$. ∎

The following easy lemma will be used in the next proposition.

[7] They used strings instead of terms to model messages, thus the decryption property was written as $\mathsf{dec}_a \cdot \mathsf{enc}_a \equiv \epsilon$, where $\equiv$ is an equivalence relation on words, dec_a, enc_a are letters, and a is just an index.

[8] except for Chapter 3 in which we use an implicit arbitrary rewriting system.

Lemma 1.15 *Let $\mathcal{R}$ be a confluent rewriting system. Then for all terms u, v such that v is in normal form, $u \to_{\mathcal{R}}^{*} v$ if and only if $u =_{\mathcal{E}(\mathcal{R})} v$.*

Proposition 1.16 *Let $\mathcal{R}$ be a simple rewriting system. Let T be a set of terms, and t be a term, with $T, t \subseteq \mathcal{T}(\mathcal{F}_{\mathsf{cstr}}, \mathcal{X}, \mathcal{N})$, such that t is in normal form. Then $T \vdash_{\mathcal{I}^d(\mathcal{R})} t$ if and only if $T \vdash_{\mathcal{I}(\mathcal{E}(\mathcal{R}))} t$.*

Proof If $T \vdash_{\mathcal{I}^d(\mathcal{R})} t$ then clearly $T \vdash_{\mathcal{I}(\mathcal{E}(\mathcal{R}))} t$, since every instance of a rule ($\mathsf{Rw}_{\mathcal{R}}$) is also an instance of the rule ($\mathsf{Eq}_{\mathcal{E}(\mathcal{R})}$) (indeed, from $u \to_{\mathcal{R}}^{*} v$ it follows that $u =_{\mathcal{E}(\mathcal{R})} v$).

Suppose now that $T \vdash_{\mathcal{I}(\mathcal{E}(\mathcal{R}))} t$. From Lemma 1.10 we have that there exists a public context C such that $C[t_1, \ldots, t_n] =_{\mathcal{E}(\mathcal{R})} t$ for some $t_i \in T$ $(0 \leq i \leq n)$. Then, since t is in normal form, from Lemma 1.15 it follows that $C[t_1, \ldots, t_n] \to_{\mathcal{R}}^{*} t$. Hence, using Lemma 1.14, we have that $T \vdash_{\mathcal{I}^d(\mathcal{R})} t$. ∎

This proposition shows that it is sufficient to use one of the deduction systems $\mathcal{I}(\mathcal{R})$ and $\mathcal{I}(\mathcal{E}(\mathcal{R}))$ as along as we are only interested in terms in normal form. And indeed we are, since we can safely suppose that all sent and received messages are only modeled by normalised terms. However, we do not make this assumption, and hence when working over the entire $\mathcal{F}$ (i.e. with explicit destructors) we use the deduction system $\mathcal{I}(\mathcal{E}(\mathcal{R}))$. Moreover, to our knowledge the deduction system $\mathcal{I}(\mathcal{E}(\mathcal{R}))$ is absent from the literature on protocol analysis.

1.2.5 The deduction problem

Given a deduction system $\mathcal{I}$, a finite set of terms T and a term u the problem of deciding whether u is deducible from T in $\mathcal{I}$ is classically known as the *intruder deduction problem*, since usually the set T of terms represents the intruders' knowledge—the set of messages that have been sent so far over the network, and u represents a secret message, or simply a message that the intruder is trying to produce. We call it simply the deduction problem, since not only the intruder but all the agents need to produce new messages from their current knowledge.

The deduction problem is polynomial in the size of T and u for $\mathcal{I}(\mathcal{R}_0)$ (see Section 1.2.3.1), and the procedure is based on the observation that a minimal proof of $T \vdash u$ in $\mathcal{I}(\mathcal{R}_0)$ uses only subterms of T and u. Indeed, at each step of the algorithm, the current set of terms is updated with the subterms that are reachable using only one deduction rule. Using the above observation it is then sufficient to check at each such step whether u belongs to the current set of terms. The algorithm saturates and stops in a number of steps proportional with the number of subterms of T and u, which is linear in the size of the problem given that a DAG-representation of terms is used.

The mentioned observation is formalised by the following lemma.

Lemma 1.17 (Locality lemma) *Let $\mathcal{R}' \subseteq \mathcal{R}_0$. Let T be a set of terms, u a term and π a minimal proof of $T \vdash_{\mathcal{I}(\mathcal{R}')} u$. Then all terms occurring in the nodes of π are in $\mathrm{st}(T, u)$. If the last rule of π is a decomposition (i.e. in $\mathcal{I}_d$) then these terms are in $\mathrm{st}(T)$.*

This property of (some) deduction systems, known as *locality*, is an instance of a more general property stated by D. McAllester [McA93] in the context of generic inference systems. The same property holds and is used for more complex deduction systems, while employing a different notion of subterm, suited for the specific deduction system under analysis. We do not give more details here about the deduction problem, since it has received an intensive treatment in more general contexts: see [CKRT03, CLS03, LLT05] for specific theories (XOR, homomorphic encryption, etc), [AC04, AC05, DLLT07] for generic theories, [CR06, ACD07] for combinations of theories, and [CDL06] for a survey.

1.3 Roles

Since protocols are sets of roles, we start by showing how roles are specified and executed. We then comment on the actual ability of an agent to execute a role. Finally we describe two standard types of roles.

1.3.1 Specification of roles

In the following, we fix an arbitrary signature $\mathcal{F}$ and an arbitrary equational theory $\mathcal{E}$.

Definition 1.18 (Role) *A* role *R is a tuple consisting of*

- *a finite set of variables $z_1, \ldots, z_k$ of sort* Id*, called* parameters*, with $k \geq 0$,*
- *a finite set $\widetilde{x}$ of variables, called* fresh items[9]*,*
- *a finite non-empty sequence of tuples $\big((u_i, E_i, v_i)\big)_{1 \leq i \leq p}$ called* sequence of instructions*; E_i are finite sets of equations, u_i, v_i are terms such that $\mathrm{var}(v_i) \subseteq \bigcup_{j \leq i} \big(\mathrm{var}(u_j) \cup \mathrm{var}(E_j)\big) \setminus \widetilde{x}$ and $\mathrm{names}(u_i, E_i, v_i) = \emptyset$.*

The role R is denoted $R(z_1, \ldots, z_k) = \nu\widetilde{x}.\, \mathsf{recv}(u_1), E_1, \mathsf{send}(v_1);\ \ldots\ ;\ \mathsf{recv}(u_p), E_p, \mathsf{send}(v_p)$.

In the above definition p is called the *length* of role R, the i-th element of the sequence is called the i-th *step* or *control-point* of role R, and u_i, v_i, and the equations in E_i, are called respectively the *"receive"* and *"send"* terms, and the *(equality) tests* of role R at step i. When the set of equations is empty we simply omit it, that is, we write $(\mathsf{recv}(u), \mathsf{send}(v))$ instead of $(\mathsf{recv}(u), \emptyset, \mathsf{send}(v))$. We denote $\mathrm{var}(R) \stackrel{\mathrm{def}}{=} \bigcup_{1 \leq i \leq p} \mathrm{var}(u_i, E_i, v_i) \setminus (\{z_1, \ldots, z_k\} \cup \widetilde{x})$.

Here, parameters stand for the agents that are participating in a role session, from the point of view of the agent playing the role. Note, however, that it is not specified which concrete agent plays the role (this will be done at the protocol level). We could have considered that the parameters also represent other data, like the symmetric long-term shared keys between agents. We have preferred to model them as $\mathsf{k}(z_i, z_j)$.

We observe that a "send" is always grouped with a "receive". This is because we suppose that if a received message is as expected then the agent playing the role instantaneously sends its response. While this behaviour is not transparent at this stage, it will become clear in the following sections.

Even if in this setting each "receive" is followed by a "send" and conversely (except for the last one), we can model protocols in which an agent waits for two messages and then performs a send, or in which an agent broadcasts a message. Indeed, we can code this ability by inserting *fake* "receive" and "send" messages which are known by everybody (using e.g. the public constant fake). This is done, for example, in Chapter 5.

1.3.2 Execution of roles

When a role is to be executed, it is first initialised, that is, its parameters are instantiated by concrete agents and its fresh items by new nonces or keys. This is formalised in the following definition.

[9] Fresh items are modeled by variables for technical reasons. We will usually instantiate them by new private data, hence the notation $\nu\widetilde{x}$ below.

Definition 1.19 (Role initialising substitution, initialised role) *Let R be a role with parameters $\mathcal{Z}$, fresh items $\widetilde{x}$ and the sequence of instructions S. A* role initialising substitution *for R is a ground substitution with $\mathrm{dom}(\sigma) = \mathcal{Z} \cup \widetilde{x}$ and such that for each $z \in \mathcal{Z}$, $z\sigma$ is a public constant (of sort Id), and for each $x \in \widetilde{x}$, $x\sigma$ is a new private constant.*

Given a role R as above, an initialised role *is the sequence of instructions $S\sigma$, where σ is a role initialising substitution for R.*

Since fresh items are instantiated by new constants, any initialised role uses different constants as fresh items. Also, we can suppose without loss of generality that in initialised roles long-term symmetric keys are represented by constants, that is, each ground term $\mathsf{k}(a, b)$ can be replaced by the constant k_{ab}.

We define formally the execution of a role by giving its operational semantics in terms of transition systems. The state of the execution of a role is given by the current instantiation of its variables and by its current control point. The only action that changes the instantiation is that of receiving a message. At this stage we are not interested from whom and to whom an agent receives and respectively sends a message, and hence we suppose that it is the "environment" who sends and respectively receives it.

Definition 1.20 (Execution of a role) *Let R be a role. An* execution *of R is a sequence*

$$(\sigma_1, i_1) \xrightarrow{m_1} \ldots \xrightarrow{m_p} (\sigma_{p+1}, i_{p+1})$$

with

- *σ_1 is a role initialising substitution of R, σ_j are ground substitutions for all $1 \leq j \leq p$,*
- *$1 = i_1 \leq i_2 \leq \cdots \leq i_{p+1} \leq l + 1$, where l is the length of R,*
- *$m_j \in \mathcal{T}(\mathcal{F})$ for all $1 \leq j \leq p$,*

and $(\sigma, i) \xrightarrow{m} (\sigma', i')$ if :

- *either there exists θ such that θ extends σ, $\mathrm{dom}(\theta) \subseteq \mathrm{dom}(\sigma) \cup \mathrm{var}(u, E)$, $m =_{\mathcal{E}} u\theta$, $t\theta =_{\mathcal{E}} t'\theta$ for all $(t \doteq t') \in E$, $\sigma' = \theta$ and $i' = i + 1$, where u, v, and E are respectively the "receive" and "send" terms, and the tests of the role R at step i; in this case, we say that m is* accepted *(by role R at step i), and we call $v\sigma'$ the* sent message*,*
- *or the previous condition does not hold, $\sigma' = \sigma$, and $i' = i$ (the state remains unchanged).*

The pairs (σ, i) are states *of R, $(\sigma, i) \xrightarrow{m} (\sigma', i')$ are* transitions *between role states, and if m is accepted then (σ', i') is the* successor *state of (σ, i). An execution is* partial *if $i_{p+1} < l + 1$, and* full *if $i_{p+1} = l + 1$.*

Remark that in fact the substitution σ in a role state (σ, p) determines all the execution of the role until the step p (except the messages that were not accepted).

1.3.3 Executable roles

Not all roles defined as above are "realistic". Indeed, to be able to implement a role in an executable program, the agent playing that role should be able to deterministically build each "send" message from the previously received messages and its initial knowledge. Hence a first notion to be formalised is that of deterministic roles.

Definition 1.21 (Deterministic role)
A role R is deterministic *if for all states of R, for each accepted message there is at most one successor state, that is, if $(\sigma, i) \xrightarrow{m} (\sigma', i+1)$ and $(\sigma, i) \xrightarrow{m} (\sigma'', i+1)$, then $\sigma' =_{\mathcal{E}} \sigma''$ (where $\sigma' =_{\mathcal{E}} \sigma''$ iff $\mathrm{dom}(\sigma') = \mathrm{dom}(\sigma'')$ and $x\sigma =_{\mathcal{E}} x\sigma'$ for all $x \in \mathrm{dom}(\sigma)$).*

The above definition says that, at each step, an agent has at most one choice when sending a message. The following example shows that there exist roles for which several choices are possible.

Exemple 1.22 *Let $(\mathsf{recv}(\langle x, y\rangle), \mathsf{send}(x))$ be a role and $\mathcal{E} = \{\langle x, \langle y, z\rangle\rangle \doteq \langle\langle x, y\rangle, z\rangle\}$ be the equational theory representing associativity. Then for $m = \langle a, \langle b, c\rangle\rangle$ there could be two sent messages: a or $\langle a, b\rangle$. That is, this role is not deterministic.*

Note that, by definition, once a message m is accepted, an agent has at least one choice when sending a message. And, while there is always a message m that matches a "receive" term u, it is not guaranteed neither that the "environment" is able to build such a m, nor that m passes the equality tests. We will deal with these problems at the level of protocols, since we are not interested yet what the "environment" can do. However, we can ask ourselves whether, once a message is accepted, the corresponding sent message is constructible by (the agent playing) the role using its current knowledge.

Definition 1.23 (Initial knowledge, constructible role)
Let $R(z_1, \ldots, z_k) = \nu\widetilde{x}.\, \mathsf{recv}(u_1), E_1, \mathsf{send}(v_1); \ldots; \mathsf{recv}(u_p), E_p, \mathsf{send}(v_p)$ be a role, and kn be a set of terms with $\mathrm{var}(\mathsf{kn}) \subseteq \widetilde{x} \cup \{z_1, \ldots, z_k\}$, called the initial knowledge *of R. The role R is* constructible *w.r.t. kn if for any state $(\sigma, i+1)$ of R, with $i \geq 1$, we have $\mathsf{kn}\sigma, u_1\sigma, \ldots, u_i\sigma \vdash_{\mathcal{I}(\mathcal{E})} v_i\sigma$.*

Usually, the initial knowledge of (an agent playing) a role consists of the identities and the public keys of its communication partners, its own private keys, and its long-term shared keys. Of course, an agent also knows the fresh items he creates. For example, in an asymmetric setting, the initial knowledge could be $\mathsf{kn} = \widetilde{x} \cup \{\mathsf{dk}(z_j), \mathsf{sk}(z_j)\} \cup \bigcup_i \{z_i\}$ where j is the index of the agent playing the role. Note that the terms in $\bigcup_i \{\mathsf{ek}(z_i), \mathsf{vk}(z_i)\}$ are deducible from kn.

For simplicity, we asked that only sent messages are constructible, while we could have also asked that "receive" terms and tests are constructible.

The two notions, deterministic and constructible roles are independent, as shown by the next example and Example 1.22.

Exemple 1.24 *Consider the role $(\mathsf{recv}(\{\!|x|\!\}_k), \mathsf{send}(x))$. This role is deterministic, but depending on whether $k \in \mathsf{kn}$, it is constructible or not. Also, the role $(\mathsf{recv}(\mathsf{h}(x)), \mathsf{send}(x))$ is deterministic, but not constructible. (Here we have considered $\mathcal{F} = \mathcal{F}_0$ and $\mathcal{E} = \mathcal{E}_0$.)*

A simple criterium for a role to be constructible is to have this property even at the specification level, that is, (using the notations of Definition 1.23) to have $\mathsf{kn}, u_1, \ldots, u_i \vdash_{\mathcal{I}(\mathcal{E})} v_i$, for all i. Testing whether a role satisfies this criterium reduces to solving the deduction problem. As we have seen (see Section 1.2.5) this is solvable in polynomial time in most of the cases of interest.

Definition 1.25 (Executable role) *We say that a role is* executable *w.r.t.* kn *if it is deterministic and constructible w.r.t.* kn.

1.3.4 Roles with matching and roles with equality tests

Recall that a role is, in fact, a program. One could write its code using either a high-level or a low-level language. While here we abstract the actual programs that are executed, one can still think of a higher level specification in which some operations are hidden, and a lower level specification in which all the (symbolic) operations are explicit. And indeed, there are mainly two kinds of models, depending on how received messages are handled. On the one hand, received messages are matched with a pattern and messages to be sent are directly obtained. On the other hand, testing for the required format of the received message and building the sent messages are done explicitly.

We consider in the following a simple rewriting system $\mathcal{R}$ over a signature $\mathcal{F}$.

Definition 1.26 (Role with matching, role with equality tests)
Let $R(z_1, \ldots, z_k) = \nu\widetilde{n}.\, \mathsf{recv}(u_1), E_1, \mathsf{send}(v_1); \ldots ; \mathsf{recv}(u_p), E_p, \mathsf{send}(v_p)$ *be a role. The role* R *is* with matching *if* $E_i = \emptyset$ *and* $u_i, v_i \in \mathcal{T}(\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}, \mathcal{X}, \mathcal{N})$ *for all* $1 \leq i \leq p$. *The role* R *is* with equality tests *if* u_i *is* x_i *and* $\mathrm{var}(E_i) \subseteq \{x_1, \ldots, x_i\}$, *for all* $1 \leq i \leq p$.

Note that for roles with equality tests, there are no new variables in the tests. An instruction with new variables in tests would actually perform some kind of pattern matching. For example, $(\mathsf{recv}(x), [x \doteq \mathsf{pair}(x_1, x_2)], \mathsf{send}(v))$ is "equivalent" with $(\mathsf{recv}(\mathsf{pair}(x_1, x_2)), \mathsf{send}(v))$.

Remark For roles with matching we assume that the signature on which we work is $\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}$. This supposes that, in the Definition 1.20, this is the signature over which the messages m (received from the environment) are built. Also, since the deduction systems $\mathcal{I}(\mathcal{E}(\mathcal{R}))$ and $\mathcal{I}(\mathcal{R})$ are equivalent over $\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}$ (see Proposition 1.11), we work in fact with the deduction system $\mathcal{I}(\mathcal{R})$. These assumptions correspond to the idea that for roles with matching the destructors operations are implicit. This remark is schematised by the following table:

type of roles	signature	deduction system
roles with matching	$\mathcal{F} \setminus \mathcal{F}_{\mathsf{dstr}}$	$\mathcal{I}(\mathcal{R})$
roles with equality tests	$\mathcal{F}$	$\mathcal{I}(\mathcal{E}(\mathcal{R}))$

For us, in the following chapters, all roles will be either with matching or with equality tests. However, "hybrid" models, with both pattern-matching and equality tests, do exist (see e.g. [CLR07]). They are useful, for example, to specify tests that cannot be performed implicitly by the pattern-matching.

In the model we have presented, both the specification of roles (the "receive", "send" terms, the tests) and the execution of roles (received messages m) use the same signature (either with, or without destructors). However, this is not the case for all symbolic models. An example is the process calculus [AB02] used as input to the verification tool ProVerif [Bla01]. There, operations on messages are explicit (thus the roles are specified using explicit destructors), but sent messages do not contain destructors since it is supposed that such terms do not represent valid messages (the normal execution is stopped if the application of a destructor "fails").

1.4 Protocols

1.4.1 Specification of protocols

A k-party protocol consists of k roles glued together with an association that maps each step of a role that expects some message m to the step of the role where the message m is produced. This association essentially defines how the execution of a protocol should proceed in the absence of the intruder.

Definition 1.27 (Protocol) *A k-party* protocol *is a pair $\Pi = (\mathcal{R}, \mathcal{S})$ where $\mathcal{R}$ is a sequence of k roles with k parameters, and $\mathcal{S} : [k] \times \mathbb{Z} \hookrightarrow [k] \times \mathbb{Z}$ is a partial mapping.*

The i-th role in $\mathcal{R}$ is denoted $\mathcal{R}_i$ and we may simply refer to a role by its index. The function $\mathcal{S}$ returns for each role/control-point pair (r, p), the role/control-point pair $(r', p') = \mathcal{S}(r, p)$ which emits the message to be processed by role r at step p.

Exemple 1.28 *The Needham-Schroeder protocol [NS78] (presented in the Introduction, page 15)*

$$\begin{array}{ll} A \Rightarrow B: & \{\![N_a, A]\!\}_{\mathsf{ek}(B)} \\ B \Rightarrow A: & \{\![N_a, N_b]\!\}_{\mathsf{ek}(A)} \\ A \Rightarrow B: & \{\![N_b]\!\}_{\mathsf{ek}(B)} \end{array}$$

is specified as follows: there are two roles $\mathcal{R}_1$ and $\mathcal{R}_2$ corresponding to the sender's role and the receiver's role. For each relevant "receive", the corresponding value of $\mathcal{S}$ is given on the same line.

$$\begin{array}{rlll} \mathcal{R}_1(z_a, z_b) = & \nu n_a. & \mathsf{recv}(\mathsf{init}), \mathsf{send}(\{\![n_a, z_a]\!\}_{\mathsf{ek}(z_b)}); & \\ & & \mathsf{recv}(\{\![n_a, x_{n_b}]\!\}_{\mathsf{ek}(z_a)}), \mathsf{send}(\{\![x_{n_b}]\!\}_{\mathsf{ek}(z_b)}). & \mathcal{S}(1,2) = (2,1) \\ \mathcal{R}_2(z_a, z_b) = & \nu n_b. & \mathsf{recv}(\{\![y_{n_a}, z_a]\!\}_{\mathsf{ek}(z_b)}), \mathsf{send}(\{\![y_{n_a}, n_b]\!\}_{\mathsf{ek}(z_a)}); & \mathcal{S}(2,1) = (1,1) \\ & & \mathsf{recv}(\{\![n_b]\!\}_{\mathsf{ek}(z_b)}), \mathsf{send}(\mathsf{stop}). & \mathcal{S}(2,2) = (1,2) \end{array}$$

The variables n_a, n_b are of sort Nonce*, and the variable x_{n_b} and y_{n_a} are of sort* Msg*. The subscripts are there for readability and have no formal meaning. Moreover, one should rather think of them as written with uppercase letters to suggest roles (or variables) and not concrete agents (or constants).*

1.4.2 Execution of protocols

The intruder As mentioned in the Introduction, the malicious environment in which a protocol is executed is represented by a special agent, the intruder. We suppose that the communication is under his complete control, and he can intercept, drop, or modify the messages on the network. This allows us to assume that all communications pass through the intruder: it is always him who sends the messages to agents playing a protocol, and him who receives the messages sent by the agents. Moreover, we assume that it is the intruder who decides when an agent starts a new (role) session, and which are the agent's communication partners in this session.

This corresponds to the intuition that transitions between two global states are caused by actions of the adversary who can initiate new sessions of the protocol between users that he chooses, and send messages to existing sessions

The intruder is also able to corrupt parties. This happens only at the beginning of an execution of a protocol (that is, we are only concerned with the case of static corruption). By corrupting an agent, the intruder obtains the agent's private information, like its secret decryption and signing key. All the data obtained by corrupting parties, and also some other data (like his own nonces, and keys, the identities of all agents, their public keys, etc.), forms the *intruder's initial knowledge*, which is represented by a set of closed terms.

Finally, we suppose that his capabilities of obtaining messages from his current knowledge are as any other's agent capabilities. This translates in using the same deduction system as agents do for executing their roles. Also, we do not restrict neither the computing power, nor the memory size of the intruder, and hence we simply ignore these aspects.

As for roles, the execution of a protocol is given in terms of transition systems. A *state* of a protocol execution is given by a triple $(\mathsf{SId}, \mathsf{f}, \mathsf{H})$. Here, SId is the set of role session ids currently executed by protocol participants, f is a global assignment function that keeps track of the local state of each existing session and H is mainly the set of messages that have been sent on the network so far.

More precisely, each role session id is a tuple of the form $(s, r, (a_1, a_2, \ldots, a_k))$, where $s \in \mathbb{N}$ is a unique identifier for the role session, r is the index of the role that is executed in the session and $a_1, a_2, \ldots, a_k \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}})$ are the identities of the parties that are involved in the session. SID denotes the set $(\mathbb{N} \times [k] \times (\mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}}))^k)$ of all session ids.

A global assignment f is a function defined on a set $\mathsf{SId} \subseteq \mathsf{SID}$ which represents the session ids initialised in the execution. For each such session id $\mathsf{sid} \in \mathsf{SId}$, $\mathsf{f}(\mathsf{sid}) = (\sigma, p)$ returns the local state of the agent playing the session.

Finally, the messages that may be sent on the network can be essentially any element of $\mathcal{T}(\mathcal{F})$.

An *execution trace* is a sequence of global states with transition between them being caused by one of the *actions* **new**, and **send** with appropriate parameters that we clarify below. This corresponds to the intuition that transitions between two global states are caused by actions of the adversary who can initiate new sessions of the protocol, and send messages to existing sessions. The formal definition follows.

Definition 1.29 (Execution trace) *For a k-party protocol* Π*, an* execution trace *is a sequence*

$$(\mathsf{SId}_0, \mathsf{f}_0, \mathsf{H}_0) \xrightarrow{\alpha_1} (\mathsf{SId}_1, \mathsf{f}_1, \mathsf{H}_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, \mathsf{f}_n, \mathsf{H}_n)$$

such that for each $0 \leq i \leq n$, $\mathsf{SId}_i \subseteq \mathsf{SID}$, $H_i \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, $\mathsf{f}_i : \mathsf{SId}_i \to \mathsf{Subst} \times \mathbb{N}$, *where* Subst *is the set of all substitutions, and the transitions* α_i *are as follows:*

- *The adversary can initiate new sessions:*

$$(\mathsf{SId}, \mathsf{f}, \mathsf{H}) \xrightarrow{\mathbf{new}(r, a_1, \ldots, a_k)} (\mathsf{SId}', \mathsf{f}', \mathsf{H})$$

  where $1 \leq r \leq k$, $a_1, \ldots, a_k \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}})$, $\mathsf{SId}' = \mathsf{SId} \cup \{\mathsf{sid}'\}$, *with* $\mathsf{sid}' = (\sharp\mathsf{SId} + 1, r, (a_1, \ldots, a_k))$*, and the function* f' *is defined by:*

  - $\mathsf{f}'(\mathsf{sid}) = \mathsf{f}(\mathsf{sid})$ *for every* $\mathsf{sid} \in \mathsf{SId}$*,*
  - $\mathsf{f}'(\mathsf{sid}') = (\sigma_0, 1)$ *where* σ_0 *is a role initialising substitution for role* r*, such that* $z_i\sigma_0 = a_i$ *where* $z_1, \ldots, z_k$ *are the parameters of role* r*.*

- *The adversary can send messages to existing sessions:*

$$(\mathsf{SId}, \mathsf{f}, \mathsf{H}) \xrightarrow{\mathbf{send}(\mathsf{sid}, m)} (\mathsf{SId}, \mathsf{f}', \mathsf{H}')$$

  where $\mathsf{sid} \in \mathsf{SId}$, $m \in \mathcal{T}(\mathcal{F})$ *such that* $\mathsf{H} \vdash m$, f' *is defined by*

– $\mathsf{f}'(\mathsf{sid}') = \mathsf{f}(\mathsf{sid}')$ *for every* $\mathsf{sid}' \in \mathsf{SId} \setminus \{\mathsf{sid}\}$,
– $\mathsf{f}'(\mathsf{sid}) = (\sigma', p')$ *where* $\mathsf{f}(\mathsf{sid}) = (\sigma, p)$ *and* $(\sigma, p) \xrightarrow{m} (\sigma', p')$ *is a transition between states of role* r,

and H' *is defined by:*

– *if* m *is accepted then* $\mathsf{H}' = \mathsf{H} \cup \{m'\}$ *where* m' *is the corresponding sent message,*
– *otherwise,* $\mathsf{H}' = \mathsf{H}$ *(the state is not changed).*

Exemple 1.30 *Playing with the Needham-Schroeder protocol described in Example 1.28, an adversary can start a new session for the second role with players* a, b, *and send the message* $\{\![\mathsf{n}_c, a]\!\}_{\mathsf{ek}(b)}$ *to the player of the second role, where* c *is a corrupted agent. The corresponding execution trace is:*

$$(\emptyset, f_1, \mathsf{kn}) \xrightarrow{\mathbf{new}(2,a,b)} (\{\mathsf{sid}_1\}, f_2, \mathsf{kn}) \xrightarrow{\mathbf{send}(\mathsf{sid}_1, \{\![\mathsf{n}_c,a]\!\}_{\mathsf{ek}(b)})} \left(\{\mathsf{sid}_1\}, f_3, \mathsf{kn} \cup \{\![\mathsf{n}_c, \mathsf{n}_b]\!\}_{\mathsf{ek}(a)}\right),$$

where $\mathsf{kn} = \{\mathsf{dk}(c)\}$, $\mathsf{sid}_1 = (1, 2, (a, b))$, *and* f_2, f_3 *are defined as follows:* $f_2(\mathsf{sid}_1) = (\sigma_1, 1)$, $f_3(\mathsf{sid}_1) = (\sigma_2, 2)$ *where* $\sigma_1(z_a) = a$, $\sigma_1(z_b) = b$, $\sigma_1(n_b) = \mathsf{n}_b$, *and* σ_2 *extends* σ_1 *by* $\sigma_2(y_{n_a}) = \mathsf{n}_c$, *with* a, b, c *public constants of sort* Id, *and* $\mathsf{n}_b, \mathsf{n}_c$ *private constants of sort* Nonce.

1.4.3 Executable protocols

Clearly, not all protocols written using the above syntax are meaningful. Indeed, not only roles need to be executable, but also the interleaving of "send"-s and "receive"-s (given by the function $\mathcal{S}$) should be realisable. This requires in particular that $\mathcal{S}$ is consistent with the specification of roles, and that there exists a *normal execution*, that is, an execution in the absence of the intruder. In our formalism this translates to an execution in which the intruder only forwards the received messages according to the function $\mathcal{S}$.

Definition 1.31 (Executable protocols)
A k-party protocol $\Pi = (\mathcal{R}, \mathcal{S})$ *is* executable *w.r.t.* $\mathsf{kn}_1, \ldots, \mathsf{kn}_k$ *if:*

- *the function* $\mathcal{S}$ *is injective;*
- *for each role* r, $\mathcal{S}(r, i)$ *is defined if* $1 \leq i \leq p_r$ *and the "receive" term* $u_i \notin \{\mathsf{init}, \mathsf{fake}\}$, *and it is undefined otherwise, where* p_r *is the length of role* r*;*
- *each role* r *is executable w.r.t.* kn_r*;*
- *there exist ground substitutions* $\sigma_1, \ldots, \sigma_r$ *such that for all roles* r

  – σ_r *extends a role initialising substitution and* $z_i^r \sigma_r = a_i$, *for all* $1 \leq i \leq p_r$,
  – $t\sigma_r =_{\mathcal{E}} t'\sigma_r$, *for all* $[t \doteq t'] \in E_i^r$,
  – $u_i^r \sigma =_{\mathcal{E}} v_{i'}^{r'} \sigma$ *for all* i *such that* $\mathcal{S}(r, i) = (r', i')$ *is defined,*

  where z_i^r *are the parameters of role* r, a_i *are pairwise different constants of sort* Id, u_i^r, $v_{i'}^{r'}$, *and* E_i^r *are respectively the* i*-th "receive" term of role* r, *the* i'*-th "send" of role* r', *and the* i*-th set of equality tests of role* r.

The substitution σ is obvious for all "realistic" protocols in the literature, and thus checking whether a protocol is executable should be very easy. In view of an automatic verification of specifications of protocols, one can either require that σ is given, or apply the obvious algorithm for obtaining σ (i.e. simulate a normal execution).

Remark that with this definition an agent may send terms which are not received by other agents (we have imposed that for each "receive" there is a (unique) associated "send", but not vice-versa, that for each "send" there is a corresponding "receive"). This definition is enough however to show that most of protocols that code an undecidable problem [DLMS99, AC02] are not executable. Moreover, it is undecidable to check whether those protocols have a normal execution (one without intruder). Let us mention that there are conjectures (see e.g. [Frö07]) stating that the secrecy problem is decidable for executable protocols using bounded-message size. On the other hand, the restrictions imposed by the definition of executability do not seem so strong in order to affect (un)decidability (considering here perfect cryptography).

In the following we will not suppose that protocols are executable, meaning that the results we present do not depend on this assumption.

Part I

Decidability results

Chapter 2

Decidability results using constraint systems. Application to key cycles

Contents

2.1 The model **56**
- 2.1.1 Constraint systems 56
- 2.1.2 From protocols to constraint systems 57
- 2.1.3 Security properties 58

2.2 Simplifying constraint systems **60**
- 2.2.1 Simplification rules 60
- 2.2.2 Decision procedure in NP-time 61
- 2.2.3 Correctness 63
- 2.2.4 Completeness 65
- 2.2.5 Termination in polynomial time 68
- 2.2.6 An alternative approach to polynomial-time termination 71

2.3 Decidability of some specialised security properties **73**
- 2.3.1 Detection of key cycles 73
- 2.3.2 Secrecy for protocols with timestamps 82

2.4 Conclusions **84**

In this chapter we re-investigate and extend the constraint system approach for a bounded number of sessions [MS01, CLS03]. We provide a generic procedure to decide general security properties by showing that any constraint system can be transformed in (possibly several) much simpler constraint systems. As a consequence, we prove that deciding the existence of key-cycles is NP-complete for a bounded number of sessions. As an other application, we give an alternative decision procedure to a significant fragment of protocols with timestamps.

Outline of the chapter The model is presented in Section 2.1, where we define constraint systems (§2.1.1) and show how they can be used to express protocol executions (§2.1.2). We also define here security properties and the notion of satisfiability of constraint systems (§2.1.3). In Section 2.2, we explain how the satisfiability problem of any security property can be non-deterministically, polynomially reduced to the satisfiability of the same problem but on simpler constraint systems. The simplification rules derived from [CLS03] (which are provided in §2.2.1) are actually not sufficient to ensure termination in polynomial time. Thus we introduce in Section 2.2.2 a refined decision procedure, which is correct, complete, and terminating in polynomial time (proofs can be found in §2.2.3, §2.2.4, and §2.2.5 respectively). An alternative approach to polynomial time termination is sketched in §2.2.6. We next show how the constraint systems approach can be used to obtain our main result of NP-completeness of the detection of key cycles (§2.3.1). We also show how it can be used to derive NP-completeness for protocols with timestamps (§2.3.2). Some concluding remarks can be found in Section 2.4.

2.1 The model

Constraint systems are quite common in modeling security protocols for a bounded number of sessions. We recall here their formalism, and show how they can be used to specify general security properties.

We fix from the start a concrete signature and deduction system used throughout this chapter, although the definitions given in this section work as well in a general setting. The sort system is arbitrary such that it contains the supersort Msg. The function symbols are those occurring in the deduction system $\mathcal{I}_0$ presented in Section 1.2.3.1 (page 40), which is also the deduction system we consider in this chapter. In fact, the rule (Retr) is optional. That is, our results hold in both cases (when the deduction relation $\vdash$ is defined with or without this rule).

We consider in this chapter only roles with matching and we will just call them roles. Constraint systems have also been used for roles with equality tests in [DJ04, Bau07].

2.1.1 Constraint systems

Definition 2.1 *A* constraint system C *is a finite set of expressions* $T \Vdash u$*, called* constraints*, where* T *is a non empty set of terms, called the* left-hand side *of the constraint and* u *is a term, called the* right-hand side *of the constraint, such that:*

- *the left-hand sides of all constraints are totally ordered by inclusion;*
- *if* $x \in \mathrm{var}(T)$ *for some* $(T \Vdash u) \in C$ *then*

$$T_x \stackrel{\mathrm{def}}{=} \min\{T' \mid (T' \Vdash u') \in C, x \in \mathrm{var}(u')\}$$

  exists and $T_x \subsetneq T$.

A solution *of* C *is a closed substitution* θ *such that for all* $(T \Vdash u) \in C$, $T\theta \vdash u\theta$.

The *left-hand side* of a constraint system C, denoted by $\mathsf{lhs}(C)$, is the maximal left-hand side of the constraints of C. The *right-hand side* of a constraint system C, denoted by $\mathsf{rhs}(C)$, is the set of right-hand sides of its constraints. $\mathrm{var}(C)$ denotes the set of variables occurring in C. $\perp$ denotes the unsatisfiable system. The *size* of a constraint system is defined as $|C| \stackrel{\mathrm{def}}{=} |\mathsf{lhs}(C) \cup \mathsf{rhs}(C)|$.

A constraint system C is usually denoted as a conjunction of constraints

$$C = \bigwedge_{1 \le i \le n} (T_i \Vdash u_i)$$

with $T_i \subseteq T_{i+1}$, for all $1 \le i \le n-1$. The second condition in Definition 2.1 then implies that if $x \in \mathrm{var}(T_i)$ then $\exists j < i$ such that $T_j = T_x$ and $T_j \subsetneq T_i$.

2.1.2 From protocols to constraint systems

We present now how to build constraint systems starting from an execution scenario (e.g. agent a plays two role sessions with b, and b one role session with A). The solutions of these constraint systems represent all possible executions of the given scenario.

Similar presentations can be found in [Cor06, Del06].

Definition 2.2 (Scenario) *A* scenario *is a finite set of initialised protocol roles.*

Exemple 2.3 *Consider again the Needham-Schroeder protocol [NS78] formalised in Example 1.28 (page 48). The following two instantiated roles represent a scenario where A starts a session with a corrupted agent I (whose private key is known to the intruder) and B is willing to answer to A:*

$$R_1 = \big(\mathsf{recv}(\mathsf{init}), \mathsf{send}(\{\![\mathsf{n}_a, a]\!\}_{\mathsf{ek}(i)}); \mathsf{recv}(\{\![\mathsf{n}_a, y_{n_b}]\!\}_{\mathsf{ek}(a)}), \mathsf{send}(\{\![y_{n_b}]\!\}_{\mathsf{ek}(i)})\big)$$
$$R_2 = \big(\mathsf{recv}(\{\![y_{n_a}, a]\!\}_{\mathsf{ek}(b)}), \mathsf{send}(\{\![y_{n_a}, \mathsf{n}_b]\!\}_{\mathsf{ek}(a)}); \mathsf{recv}(\{\![\mathsf{n}_b]\!\}_{\mathsf{ek}(b)}), \mathsf{send}(\mathsf{stop})\big)$$

That is, the two initialising substitutions are $\sigma_1 = \{\,{}^{a}/_{z_a}, {}^{i}/_{z_b}, {}^{\mathsf{n}_a}/_{x_{n_a}}\}$ and $\sigma_2 = \{\,{}^{a}/_{z_a}, {}^{b}/_{z_b}, {}^{\mathsf{n}_b}/_{x_{n_b}}\}$.

Given a scenario, there are many ways in which the instructions of the participating roles can be interleaved in order to obtain a sequence of instructions (i.e. a possible execution).

Definition 2.4 (Interleaving) *Let $\mathsf{Sc} = \{R_1, \ldots, R_n\}$ be a scenario. An* interleaving *of S of length l is a function $\iota : [l] \to S$ such that for all $R \in S$, $\sharp\{j \mid \iota(j) = R\} \leq k_R$, where k_R is the length of the role R. We define $\kappa : [l] \to \mathbb{N}$ by $\kappa(j) = \sharp\{j' \mid \iota(j') = \iota(j), j' \leq j\}$.*

The function ι tells which is the role currently (i.e. at index j) playing, while the function κ tells which is current control-point in the role. Note that in an interleaving not all roles need to be represented and roles need not reach their final control-point.

From interleavings to constraint systems Let Sc be a scenario and ι be an interleaving of length l of Sc. We suppose that roles in Sc use different variables, that is $\mathrm{var}(r) \cap \mathrm{var}(r') = \emptyset$, for all $r, r' \in \mathsf{Sc}$ (this can be achieved by renaming the concerned variables). We denote by rcv_r^p and snt_r^p the p-th (initialised) "receive" and respectively "send" message of role $r \in \mathsf{Sc}$. Then the constraint system associated with the interleaving ι, and with the initial intruder knowledge T_0 is

$$C = \bigwedge_{1 \leq i < l} (T_i \Vdash t_i)$$

where for all $i \geq 1$,

$$T_i = T_{i-1} \cup \{\mathsf{snt}_r^p\}$$
$$t_i = \mathsf{rcv}_{r'}^{p'}$$

with $r = \iota(i)$, $p = \kappa(i)$, $r' = \iota(i+1)$, and $p' = \kappa(i+1)$.

The sets of constraints C built above is a constraint system. Indeed, the left-hand sides of constraints in C are ordered by inclusion. And, if $x \in \mathrm{var}(T_i)$ then there is a $v \in T_i$ such that $x \in \mathrm{var}(v)$. Let j be the index such that $v \in T_j \subseteq T_{j-1}$ (such a $i \geq j \geq 1$ exists by construction of C, and since T_0 is a set of ground terms). Then $v = \mathsf{snt}_r^p$ where $r = \iota(j)$ and $p = \kappa(j)$. By the definition of roles, there is $p' \leq p$ such that $x \in \mathrm{var}(\mathsf{rcv}_r^{p'})$. But, by construction of C, $\mathsf{rcv}_r^{p'} = t_{j'}$ with $j' < j$. Hence T_x exists and $T_x \subseteq T_{j'} \subsetneq T_j \subseteq T_i$.

Note that we can safely (i.e. without changing the solution set) eliminate the constraints $T_i \Vdash t_i$ with $t_i \in \{\mathsf{init}, \mathsf{stop}\}$, since these are public constants (hence always deducible). Also, in order to remember the last message sent on the network (i.e. snt_r^p with $\iota(l) = r$ and $\kappa(l) = p$), which does not appear in $\mathsf{lhs}(C)$, we add to C the constraint $T_{l-1} \cup \{\mathsf{snt}_r^p\} \Vdash c$, where c is some public constant. We call this new constraint system the *extended* constraint system associated with ι.

Exemple 2.5 *Consider the scenario presented in Example 2.3 and the interleaving ι of length 4 given by $\iota(1) = R_1$, $\iota(2) = R_2$, $\iota(3) = R_1$, and $\iota(4) = R_2$. The constraint system C_1 associated with the interleaving ι and the initial intruder knowledge $T_0 = \{a, b, i, \mathsf{ek}(a), \mathsf{ek}(b), \mathsf{ek}(i), \mathsf{dk}(i)\}$ is:*

$$T_1 \stackrel{\text{def}}{=} T_0, \{\![\mathsf{n}_a, a]\!\}_{\mathsf{ek}(i)} \quad \Vdash \quad \{\![y_{n_a}, a]\!\}_{\mathsf{ek}(b)} \tag{2.1}$$

$$T_2 \stackrel{\text{def}}{=} T_1, \{\![y_{n_a}, \mathsf{n}_b]\!\}_{\mathsf{ek}(a)} \quad \Vdash \quad \{\![\mathsf{n}_a, y_{n_b}]\!\}_{\mathsf{ek}(a)} \tag{2.2}$$

$$T_3 \stackrel{\text{def}}{=} T_2, \{\![y_{n_b}]\!\}_{\mathsf{ek}(i)} \quad \Vdash \quad \{\![\mathsf{n}_b]\!\}_{\mathsf{ek}(b)} \tag{2.3}$$

The set T_1 represents the messages known to the intruder once A has contacted the corrupted agent I. Then the equations 2.1 and 2.2 can be read as follows: if a message of the form $\{\![y_{n_a}, a]\!\}_{\mathsf{ek}(b)}$ can be obtained by the intruder, then this message would be send to B, and B would answer to this message by $\{\![y_{n_a}, \mathsf{n}_b]\!\}_{\mathsf{ek}(a)}$, which is added to T_1. Subsequently, if a message of the form $\{\![\mathsf{n}_a, y_{n_b}]\!\}_{\mathsf{ek}(a)}$ can be obtained by the intruder, then this message would be send to A, and A would answer with $\{\![y_{n_b}]\!\}_{\mathsf{ek}(i)}$ since A believes she is talking to I. The run is successful if B can finish his session by receiving the message $\{\![\mathsf{n}_b]\!\}_{\mathsf{ek}(b)}$. Then B believes he has talked to A while A actually talked to I. The variables represent those parts of messages that are a priori unknown to the agents.

2.1.3 Security properties

We are concerned here with trace properties, that is with properties that can be expressed as predicates on traces. A protocol satisfies such a property if and only if the corresponding predicate holds for each execution trace of the protocol. To verify whether a property is satisfied for a *"bounded number of sessions"* (i.e. a scenario), it is then sufficient to check whether the interleavings of the scenario represent indeed execution traces satisfying the property. Not all interleavings need to be enumerated (see [MS01, Cor06]). But anyhow, here we are only interested whether an arbitrary interleaving represents a trace satisfying a certain property. Then, in this context, we consider that a security property is just a predicate on lists of messages, since checking whether this list represents a trace is done by deciding the satisfiability of the associated (extended) constraint system.

Definition 2.6 *Let C be a constraint system, L a list of terms such that* $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$ *and P a predicate on lists of terms. A* solution *of C for P w.r.t. L is a closed substitution θ such that $\forall (T \Vdash u) \in C$, $T\theta \vdash u\theta$ and $P(L\theta)$ holds.*

Remark that a substitution is a solution of C for the **true** predicate (that holds for any list of terms) w.r.t. an arbitrary list if and only if it is a solution of C (in the sense of Definition 2.1). To avoid confusion, in such cases we call a solution of C a *partial* solution of C, or we explicitly mention the predicate **true**.

For a list L we denote by L_s the set of terms of the list L. For a predicate P we denote by $\overline{P}$ the negation of P.

The approach presented above doesn't allow to prove the correctness of a protocol w.r.t. a property (i.e. a predicate) P (one cannot check in this way all scenarios), and represents in fact a search for attacks w.r.t. a property P. That is, we are interested in the existence of attacks—in the predicate $\overline{P}$, the correctness property being thus expressed as the predicate $\overline{\overline{P}}$.

We show next how secrecy and authentication are modeled in this setting.

Secrecy This property can be easily expressed by requiring that the secret data s is not deducible from the messages sent on the network. We define the predicate P_{s} to hold on a list of messages if and only if s is deducible from it. That is, $P_{\mathsf{s}}(L)$ holds if and only if $L_s \vdash \mathsf{s}$. The secrecy property is then represented by the predicate $\overline{P}_{\mathsf{s}}$. The list L on which this predicate is usually evaluated is a list of the terms in $\mathsf{lhs}(C)$, where C is the (extended) constraint system C associated with the interleaving under study. Hence, such a deduction-based property can be directly encoded by adding the constraint $\mathsf{lhs}(C) \Vdash \mathsf{s}$ to C, and asking for the partial satisfiability of the new constraint system. Considering that the only sort is Msg, we retrieve the usual constraint system deduction problem, which is known to be NP-complete [RT03].

Exemple 2.7 *We consider again the constraint system C_1 defined in Example 2.5. Let L_1 be a list of the messages in $\mathsf{lhs}(C_1)$. Then the substitution $\sigma_1 = \{ {}^{\mathsf{n}_a}/_{y_{n_a}}, {}^{\mathsf{n}_b}/_{y_{n_b}} \}$ is a solution of C_1 for the property P_{n_b} w.r.t. L_1 and corresponds to the attack found by G. Lowe [Low96].*

Authentication This property can also be defined using a predicate P_{auth} on lists of messages. For this purpose we use correspondence assertions and we introduce, following the syntax of Avispa [ABB+05], two new private function symbols witness and request of arity 4 with the following intuition: $\text{request}(a, b, id, m)$ says that the agent a *now believes* that it is really agent b who sent the message m (that is, a authenticates b on m), and $\text{witness}(b, a, id, m)$ says that b has just sent the message m to a. The symbol id is simply a constant identifying the request since there might be several authentication goals for one protocol (e.g. the Needham-Schroeder protocol is a mutual authentication protocol, hence it has 2 authentication goals). The predicate $\overline{P}_{auth}$ holds on a list L of messages if whenever $\text{request}(a, b, id, m)$ appears in the list there is a corresponding occurrence $\text{witness}(b, a, id, m)$ (defining an injection) appearing before it in the list (that is, at a smaller position), for any agents a and b. These "status events" (i.e. terms of the form $f(z, z', id, t)$ with $f \in \{\text{witness}, \text{request}\}$, z, z' parameters, id a constant, and t a term) are in fact part of the specification of a protocol (that is, each step of a role has an associated set of such events), and a list of events is generated in the same manner as constraints systems are[10]. Then the predicate $\overline{P}_{auth}$ applied on a list L built in this way represents Lowe's definition of injective agreement [Low97]. Thus, an interleaving has an attack on the authentication property if and only if the associated constraint system C has a solution for P_{auth} w.r.t. L.

Exemple 2.8 *We consider again the constraint system C_1 defined in Example 2.5. We consider here only the authentication of A by B on N_b. The corresponding list is $L_2 = \big(\text{witness}(a, i, 2, y_{n_b}), \text{request}(b, a, 2, \mathsf{n}_b)\big)$, that is, agent a acknowledges that he sent y_{n_b} to agent i, and agent b, at the end of its role execution (thus, after receiving his nonce n_b), believes he talked with agent a. The substitution σ_1 defined in Example 2.7 is a solution of C_1 for the property P_{auth} w.r.t. L_2, since there is no corresponding witness assertion for $\text{request}(b, a, 2, \mathsf{n}_b)$ in L_2.*

In Section 2.3, we provide other examples of predicates which encode time constraints, or express that no key cycles are allowed.

[10] In fact we also assume that requests are not emitted by or for corrupted agents.

$$
\begin{array}{lrll}
R_1 & C \wedge T \Vdash u \rightsquigarrow & C & \text{if } T \cup \{\, x \mid (T' \Vdash x) \in C, T' \subsetneq T \} \vdash u \\
R_2 & C \wedge T \Vdash u \rightsquigarrow_\sigma & C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathrm{mgu}(t,u),\ t \in \mathrm{st}(T), \\
 & & & \qquad t \neq u,\ t,u \text{ not variables} \\
R_3 & C \wedge T \Vdash u \rightsquigarrow_\sigma & C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathrm{mgu}(t_1,t_2),\ t_1,t_2 \in \mathrm{st}(T), \\
 & & & \qquad t_1 \neq t_2,\ t_1,t_2 \text{ not variables} \\
R_4 & C \wedge T \Vdash u \rightsquigarrow & \bot & \text{if } \mathrm{var}(T,u) = \emptyset \text{ and } T \not\vdash u \\
R_f & C \wedge T \Vdash f(u,v) \rightsquigarrow & C \wedge T \Vdash u \wedge T \Vdash v & \text{for } f \in \mathcal{F}_{\mathsf{pub}}, f \text{ not a constant}
\end{array}
$$

Figure 2.1: Simplification rules.

2.2 Simplifying constraint systems

Using some simplification rules, solving general constraint systems can be reduced to solving simpler constraint systems that we called solved. One nice property of the transformation is that it works for any security property.

We say that a constraint system is *solved* if it is different from $\bot$ and each of its constraints are of the form $T \Vdash x$, where x is a variable. Note that the empty constraint system is solved. This corresponds to the notion of solved form in [CLS03].

Solved constraint systems with the single sort Msg are particularly simple in the case of the **true** predicate since they always have a solution, as noticed in [MS01]. 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 variables. Let $t \in T_1$. Then the substitution θ defined by $x\theta = t$ for every variable x is a solution, since $T \vdash t$ for any constraint $T \Vdash x$ of the solved system.

2.2.1 Simplification rules

The *simplification rules* we consider are defined in Figure 2.1. All the rules are in fact indexed by a substitution: when there is no index then the identity substitution is implicitly considered. We write $C \rightsquigarrow^n_\sigma C'$ if there are $C_1, \ldots, C_n$ with $n \geq 1$, $C' = C_n$, $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} C_n$ and $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. We write $C \rightsquigarrow^*_\sigma C'$ if $C \rightsquigarrow^n_\sigma C'$ for some $n \geq 1$, or if $C' = C$ and σ is the empty substitution.

Exemple 2.9 *Let us consider the following constraint system C:*

$$
\left\{ \begin{array}{lll} T_1 & \Vdash & \langle\, \{\![x]\!\}_{\mathsf{ek}(a)},\ \{\![y]\!\}_{\mathsf{ek}(a)} \,\rangle \\ T_2 & \Vdash & k_1 \end{array} \right.
$$

where $T_1 = \{\, a,\ \mathsf{ek}(a),\ \langle\, \{\![k_1]\!\}_{\mathsf{ek}(a)},\ \{\![k_2]\!\}_{\mathsf{ek}(a)} \,\rangle \,\}$ and $T_2 = T_1 \cup \{\, \{\!\{y\}\!\}_x \}$. The constraint system C can be simplified into a solved constraint system using (for example) the following sequence of simplification rules.

$$
C \overset{R_{\langle\rangle}}{\rightsquigarrow} \left\{ \begin{array}{lll} T_1 & \Vdash & \{\![x]\!\}_{\mathsf{ek}(a)} \\ T_1 & \Vdash & \{\![y]\!\}_{\mathsf{ek}(a)} \\ T_2 & \Vdash & k_1 \end{array} \right. \overset{R_{\mathsf{encad}}}{\rightsquigarrow} \left\{ \begin{array}{lll} T_1 & \Vdash & x \\ T_1 & \Vdash & \mathsf{ek}(a) \\ T_1 & \Vdash & \{\![y]\!\}_{\mathsf{ek}(a)} \\ T_2 & \Vdash & k_1 \end{array} \right. \overset{R_1}{\rightsquigarrow} \left\{ \begin{array}{lll} T_1 & \Vdash & x \\ T_1 & \Vdash & \{\![y]\!\}_{\mathsf{ek}(a)} \\ T_2 & \Vdash & k_1 \end{array} \right.
$$

since $T_1 \vdash \mathsf{ek}(a)$. *Let* $\sigma = \mathrm{mgu}\left(\{\![k_1]\!\}_{\mathsf{ek}(a)},\ \{\![y]\!\}_{\mathsf{ek}(a)}\right) = \{{}^{k_1}/_{y}\}$. *We have*

$$\left\{\begin{array}{lll} T_1 & \Vdash & x \\ T_1 & \Vdash & \{\![y]\!\}_{\mathsf{ek}(a)} \\ T_2 & \Vdash & k_1 \end{array}\right. \overset{R_2}{\rightsquigarrow_\sigma} \left\{\begin{array}{lll} T_1 & \Vdash & x \\ T_1 & \Vdash & \{\![k_1]\!\}_{\mathsf{ek}(a)} \\ T_2\sigma & \Vdash & k_1 \end{array}\right. \overset{R_1}{\rightsquigarrow} \left\{\begin{array}{lll} T_1 & \Vdash & x \\ T_2\sigma & \Vdash & k_1 \end{array}\right. \overset{R_1}{\rightsquigarrow} T_1 \ \Vdash \ x$$

since $T_1 \vdash \{\![k_1]\!\}_{\mathsf{ek}(a)}$ *and* $T_2\sigma \cup \{x\} \vdash k_1$. *Intuitively, it means that any substitution of the form* $\{{}^{m}/_{x},\ {}^{k_1}/_{y}\}$ *such that* m *is deducible from* T_1 *is solution of* C *(for the* **true** *property).*

We are intersted in simplification rules that are correct and complete, meaning that a constraint system C has a solution for a security property P if and only if there exists a constraint system C' in solved form such that $C \rightsquigarrow^*_\sigma C'$ and C' has a solution for P. Note that several simplification rules can possibly be applied to a given constraint system.

In order to obtain completeness for the above set of simplification rules we need to impose a syntactic restriction on the form of constraint systems. Indeed, as observed by H. Comon-Lundh, the following constraint system $C = (T \Vdash x) \ \wedge \ (T \cup \{\![u]\!\}_x \Vdash u)$, where $T = \{\mathsf{ek}(a), \mathsf{dk}(a)\}$ represents a counter-example for completeness: the substitution $\{{}^{\mathsf{ek}(a)}/_{x}\}$ is a partial solution of C, but C has no solved form.

We say that a constraint system C is *well-formed* if, for any subterms $\{\![u]\!\}_v$ and $\mathsf{dk}(w)$ of C, we have that v and w are ground terms.

Note that this is not a restriction in our setting (where constraint systems are obtained from protocols as in Section 2.1.2), since scenarios only consist of initialised roles, and since in any protocol in which a variable x is used to represent a public or a private key can be rewritten "equivalently" by replacing x with $\mathsf{ek}(y)$ or $\mathsf{dk}(y)$ respectively. Also, the above simplification rules preserve well-formedness: indeed, it is easily seen that, if C is well-formed and $C \rightsquigarrow_\sigma C'$, then C' is well-formed.

In what follows we will only consider well-formed constraint systems.

Theorem 2.10 *Let* C *be a constraint system,* θ *a substitution,* P *a security property and* L *a list of messages such that* $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$.

1. *(Correctness) If* $C \rightsquigarrow^*_\sigma C'$ *for some constraint system* C' *and some substitution* σ *and if* θ *is a solution of* C' *for the property* P *w.r.t.* $L\sigma$ *then* $\sigma\theta$ *is a solution of* C *for the property* P *w.r.t.* L.

2. *(Completeness) If* θ *is a solution of* C *for the property* P *w.r.t.* L*, then there exist a constraint system* C' *in solved form and substitutions* σ, θ' *such that* $\theta = \sigma\theta'$*,* $C \rightsquigarrow^*_\sigma C'$ *and* θ' *is a solution of* C' *for the property* P *w.r.t.* $L\sigma$.

As we will see in the next section we need a slight extension of the simplification rules in order to obtain their termination in polynomial time. Theorem 2.10 is proved in Sections 2.2.3 and 2.2.4. The proof is a simple extension of the proof provided in [CL04] to sorted messages and general security properties.

2.2.2 Decision procedure in NP-time

Theorem 2.10 does not suffice to ensure that deciding security properties is in NP (provided that we can decide them easily, i.e. in polynomial time, on solved constraint systems). In fact, applying the simplification rules may lead to branches of exponential length (in the size of the constraint system). Indeed when applying a simplification rule to a constraint, the initial constraint is suppressed from the constraint system and replaced by new constraint(s). But this constraint may appear again later on, due to other simplification rules. It is the case for example when considering the following constraint system.

$$\begin{array}{rcl} T_0 \stackrel{\text{def}}{=} \{\{\!\{a\}\!\}_{k_0}\} & \Vdash & \{\!\{x_0\}\!\}_{k_0} \\ T_1 \stackrel{\text{def}}{=} T_0, \{\!\{\langle x_0, \langle x_0, a\rangle\rangle\}\!\}_{k_1} & \Vdash & \{\!\{x_1\}\!\}_{k_1} \\ & \vdots & \\ T_n \stackrel{\text{def}}{=} T_{n-1}, \{\!\{\langle x_{n-1}, \langle x_{n-1}, a\rangle\rangle\}\!\}_{k_n} & \Vdash & \{\!\{x_n\}\!\}_{k_n} \\ T_{n+1} \stackrel{\text{def}}{=} T_n, a & \Vdash & x_n \end{array}$$

The constraint system C is clearly satisfiable and its size is linear in n. We have that

$$C \rightsquigarrow_{\sigma}^{2n} \left\{ \begin{array}{rcl} T_0 & \Vdash & \{\!\{x_0\}\!\}_{k_0} \\ T_{n+1}\sigma & \Vdash & x_n\sigma \end{array} \right.$$

with $\sigma(x_{i+1}) = \langle x_i, \langle x_i, a\rangle\rangle$ for $0 \leq i \leq n-1$. This derivation was obtained by applying rule R_2 and then R_1 for each constraint $T_i \Vdash \{\!\{x_i\}\!\}_{k_i}$ with $1 \leq i \leq n$. The rule R_1 cannot be applied to $T_{n+1}\sigma \Vdash x_n\sigma$ since x_0 and the keys k_i are not present in or derivable from $T_{n+1}\sigma$. Note that $\sigma' = \sigma \cup \{{}^{a}/_{x_0}\}$ is a solution of C and can be easily obtained by rule R_2 on the first constraint and then rule R_1 on both constraints.

However, there is a branch of length $3(2^n - 1)$ from $T \Vdash x_n\sigma$ leading to $T \Vdash x_0$ (in solved form), where T denotes $T_{n+1}\sigma$. This is easy to see by induction on n. It is true for $n = 0$. Then using only the rules R_{pair} and R_1, we have

$$T \Vdash x_n\sigma \stackrel{R_{\langle\rangle}}{\rightsquigarrow} \left\{ \begin{array}{rcl} T & \Vdash & x_{n-1}\sigma \\ T & \Vdash & \langle x_{n-1}\sigma, a\rangle \end{array} \right. \rightsquigarrow^m \left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & \langle x_{n-1}\sigma, a\rangle \end{array} \right. \stackrel{R_{\langle\rangle}}{\rightsquigarrow} \left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & x_{n-1}\sigma \\ T & \Vdash & a \end{array} \right. \stackrel{R_1}{\rightsquigarrow}$$

$$\stackrel{R_1}{\rightsquigarrow} \left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & x_{n-1}\sigma \end{array} \right. \rightsquigarrow^m T \Vdash x_0$$

with $m = 3(2^{n-1} - 1)$ by induction hypothesis. The length of the branch is $2 \times 3(2^{n-1} - 1) + 3 = 3(2^n - 1)$. This shows that there exist branches of exponential length in the size of the constraint.

We can prove (see Section 2.2.5) that it is actually not useful to consider constraints that have already been seen before (like the constraint $T \Vdash x_{n-1}\sigma$ in our example). Thus we store the constraints that have already been visited. Starting from the initial set of simplification rules $\mathcal{R}$, we construct a new set of simplification rules $\mathcal{R}'$. For each simplification rule $C \rightsquigarrow_\sigma C'$ in $\mathcal{R}$ we introduce in $\mathcal{R}'$ the rule

$$C; D \;\rightsquigarrow_\sigma\; C' \setminus D; (C \setminus C') \cup D$$

The constraints in D are those which were already analysed, they are stored in D. The initial constraint system has the form $C; \emptyset$.

Theorem 2.11 *Let C be a constraint system, θ a substitution, P a security property and L a list of messages such that* $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$.

1. *(Correctness) If $C; \emptyset \rightsquigarrow_\sigma^n C'; D'$ for some constraint system C' and some substitution σ, if θ is a solution of C' for the property P w.r.t. $L\sigma$ then $\sigma\theta$ is a solution of C for the property P w.r.t. L.*

2. *(Completeness) If θ is a solution of C for the property P w.r.t. L, then there exist a constraint system C' in solved form, a set of constraints D' and substitutions σ, θ' such that $\theta = \sigma\theta'$, $C; \emptyset \rightsquigarrow_\sigma^* C'; D'$ and θ' is a solution of C' for the property P w.r.t. $L\sigma$.*

3. *(Termination) If $C; \emptyset \rightsquigarrow_\sigma^n C'; D'$ for some constraint system C' and some substitution σ then n is polynomially bounded in the size of C.*

The following corollary is easily obtained from the previous theorem by observing that we can guess the simplification rules which lead to a solved form.

Corollary 2.12 *Any property P that can be decided in polynomial time on solved constraint systems can be decided in non-deterministic polynomial time on arbitrary constraint systems.*

The rest of Section 2.2 is devoted to the proof of the two theorems. We first show correctness (in Section 2.2.3) and completeness (in Section 2.2.4) of Theorem 2.10. Then we prove Theorem 2.11 in Section 2.2.5.

2.2.3 Correctness

We first give two simple lemmas.

Lemma 2.13 *If $T \vdash u$ then* $\mathrm{var}(u) \subseteq \mathrm{var}(T)$.

Proof The affirmation follows easily by induction on the depth of a proof of $T \vdash u$, observing that no deduction rule in $\mathcal{I}_0$ introduces new variables; that is, $\mathrm{var}(t) \subseteq \bigcup_i \mathrm{var}(t_i)$ for deduction rules

$$\frac{S \vdash t_1 \quad \ldots \quad S \vdash t_k}{S \vdash t}$$

(without conditions), and $\mathrm{var}(t) \subseteq \mathrm{var}(T)$ if $t \in T$ (that is, for the membership rule). ∎

The next lemma shows the "cut elimination" property for the deduction system $\vdash$.

Lemma 2.14 *If $T \vdash u$ and $T, u \vdash v$ then $T \vdash v$.*

Proof Consider a proof π of $T \vdash u$ and a proof π' of $T, u \vdash v$. The tree obtained from π' by

- replacing the labels of nodes $T, u \vdash t$ in π' with $T \vdash t$,
- replacing each new leaf $T \vdash u$ (the old $T, u \vdash u$) with the tree π,

is a proof of $T \vdash v$. ∎

As a consequence, we have that if $T \subseteq T'$, $T' \vdash v$ and $T \vdash u$, for all $u \in T' \setminus T$, then $T \vdash v$.
We show now that the simplification rules preserve constraint systems.

Lemma 2.15 *The simplification rules transform a constraint system into a constraint system.*

Proof Suppose that C is a constraint system, $C = \bigwedge_i (T_i \Vdash u_i)$ and $C \rightsquigarrow_\sigma C'$. Since $T_i \subseteq T_{i+1}$ implies $T_i\sigma \subseteq T_{i+1}\sigma$ we have that C' meets point 1 of the definition of constraint systems.

We show that C' also meets point 2 of the definition of constraint systems. Let $(T' \Vdash u') \in C'$ and $x \in \mathrm{var}(T')$. We have to prove that T'_x exists and $T'_x \subsetneq T'$. We consider which simplification rule was applied.

- If rule R_1 was applied, eliminating the constraint $T \Vdash u$, then u is ground, $(T' \Vdash u') \in C$ and
$$\begin{aligned} T'_x &= \min\{T'' \mid (T'' \Vdash u'') \in C', x \in \mathrm{var}(u'')\} \\ &= \min\{T'' \mid (T'' \Vdash u'') \in C \setminus \{T \Vdash u\}, x \in \mathrm{var}(u'')\}. \end{aligned}$$
(The first equality is the definition of T'_x, and the second one holds since u is ground.) If $\min\{T'' \mid (T'' \Vdash u'') \in C \setminus \{T \Vdash u\}, x \in \mathrm{var}(u'')\} = \min\{T'' \mid (T'' \Vdash u'') \in C, x \in \mathrm{var}(u'')\}$ then the property is clearly satisfied (as it is satisfied by $T' \Vdash u'$ in C). Otherwise, we have that $x \in \mathrm{var}(u)$ and $T = \min\{T'' \mid (T'' \Vdash u'') \in C, x \in \mathrm{var}(u'')\}$. By minimality of T it follows that $x \notin \mathrm{var}(T)$ and $x \notin \{y \mid (T'' \Vdash y) \in C, T'' \subsetneq T\}$. Since $x \in \mathrm{var}(u)$, we have that $T \cup \{y \mid (T'' \Vdash y) \in C, T'' \subsetneq T\} \nvdash u$ which contradicts the applicability of rule R_1 (by Lemma 2.13).

- If rules R_2 or R_3 were applied then for each constraint $(T'' \Vdash u'') \in C'$ there is a constraint $(T \Vdash u) \in C$ such that $T\sigma = T''$ and $u\sigma = u''$. Consider $(T \Vdash u) \in C$ such that $T\sigma = T'$ and $u\sigma = u'$.

  If x is not introduced by σ then $x \in \mathrm{var}(T)$. Then T_x exists and $T_x \subsetneq T$. Thus $T_x\sigma \subseteq T\sigma$. If $T_x\sigma = T\sigma$ then $x \in \mathrm{var}(T_x)$ which contradicts the minimality of T_x. Thus $T_x\sigma \subsetneq T\sigma$. We also have that $\{T''\sigma \mid (T'' \Vdash u'') \in C, x \in \mathrm{var}(u'')\} \subseteq \{T''\sigma \mid (T''\sigma \Vdash u''\sigma) \in C', x \in \mathrm{var}(u''\sigma)\}$, since for any term u'', if $x \in \mathrm{var}(u'')$ then $x \in \mathrm{var}(u''\sigma)$. It follows that T'_x exists and $T'_x \subseteq T_x\sigma$. Hence $T'_x \subsetneq T'$.

  Otherwise, suppose x is introduced by σ, that is $\exists y \in \mathrm{var}(T)$ such that $x \in \mathrm{var}(y\sigma)$. Then T_y exists and $T_y \subsetneq T$. We choose y such that T_y is minimal with respect to the inclusion relation. We have that $T'_x \subseteq \min\{T''\sigma \mid (T'' \Vdash u'') \in C, z \in \mathrm{var}(u''), x \in \mathrm{var}(z\sigma)\} \subseteq T_y\sigma$. Again from $T_y \subsetneq T$ we obtain that $T_y\sigma \subsetneq T\sigma$, since if $T_y\sigma = T\sigma$ there exists $z \in \mathrm{var}(T_y)$ such that $x \in \mathrm{var}(z\sigma)$. We have $z \neq y$ by minimality of T_y. Thus there exists $T_z \subsetneq T_y$, which contradicts the minimality of y. Hence T'_x exists and $T'_x \subsetneq T'$.

- If rule R_4 was applied then the obtained result is a constraint system by definition.

- If rule R_f was applied, then the property is preserved, since, if $x \in \mathrm{var}(u'')$, for some term u'' such that $(T'' \Vdash u'') \in C'$, then there is a term v with $x \in \mathrm{var}(v)$ such that $(T'' \Vdash v) \in C$.

■

Lemma 2.16 (correctness) *If $C \rightsquigarrow_\sigma C'$ then for every solution τ of C' for the property P w.r.t $L\sigma$, $\sigma\tau$ is a solution of C for the property P w.r.t. L.*

Proof If the applied rule was R_1 then we have to prove that $T\tau \vdash u\tau$, where $T \Vdash u$ is the eliminated constraint. We know that $T \cup \{x \mid T' \Vdash x \in C, T' \subsetneq T\} \vdash u$. It follows that $T\tau \cup \{x\tau \mid T' \Vdash x \in C, T' \subsetneq T\} \vdash u\tau$. For any $(T' \Vdash x) \in C, T' \subsetneq T$, we have that $T'\tau \vdash x\tau$, and hence $T\tau \vdash x\tau$. Then from Lemma 2.14 we obtain that $T\tau \vdash u\tau$.

Suppose that the rule applied in order to obtain C' was R_2 or R_3. Then we have for each constraint $T \Vdash u$ of C that $(T\sigma)\tau \vdash (u\sigma)\tau$, that is, $T(\sigma\tau) \vdash u(\sigma\tau)$. If the rule was R_f then we obtain that $T\tau \vdash f(u, v)\tau$ from $T\tau \vdash u\tau$ and $T\tau \vdash v\tau$ by applying the corresponding rule (e.g. encryption if $f = \mathsf{encd}$). Finally, rule R_4 couldn't have been applied.

We deduce that $\sigma\tau$ satisfies C. Moreover, since τ is solution of C' for the property P w.r.t $L\sigma$, it means that $P((L\sigma)\tau)$ holds, that is $P(L(\sigma\tau))$ holds. Thus $\sigma\tau$ is solution of C for the property P w.r.t L. ■

2.2.4 Completeness

Let $T_1 \subseteq T_2 \subseteq \cdots \subseteq T_n$. We say that a proof π of $T_i \vdash u$ is *left minimal* if for any $j < i$ such that $T_j \vdash u$, π' is a proof of $T_j \vdash u$ where π' is obtained from π by replacing T_i with T_j in the left-hand side of each node of π. We say that a proof is *simple* if any subproof is left minimal and on any branch there are no two equal nodes. Remark that a subproof of a simple proof is simple.

Lemma 2.17 *If $T_i \vdash u$ then there is a simple proof of it.*

Proof We prove the property by induction on the pair (i, m) (considering the lexicographical order), where m is the size of a proof of $T_i \vdash u$.

If $i = 1$ then any (subproof of any) proof of $T_1 \vdash u$ is left minimal and there exists a proof without repeating nodes on branches.

If $i > 1$ and there is $j < i$ such that $T_j \vdash u$ then apply the recursion hypothesis to obtain the existence of a simple proof of $T_j \vdash u$. This proof is also a simple proof of $T_i \vdash u$ (by using weakening; i.e. if $T \vdash u$ and $T \subseteq T'$ then $T' \vdash u$).

If $i > 1$ and there is no $j < i$ such that $T_j \vdash u$, then apply the recursion hypothesis on the immediate subproofs of the proof of $T_i \vdash u$. If the node $T_i \vdash u$ appears in one of the obtained subproofs π' then consider a proof of $T_i \vdash u$ (subproof of π') not having $T_i \vdash u$ as an internal node. Otherwise apply the same last rule to obtain the root node $T_i \vdash u$. Anyhow, the obtained proof and all of its subproofs are left minimal by construction, and hence the obtained proof is simple. ∎

For a constraint system C, we call a left-hand side T_i of some constraint in C *minimal unsolved* if for all $(T \Vdash u) \in C$ such that $T \subsetneq T_i$, u is a variable, and there is a constraint $T_i \Vdash u_i$ with u_i not a variable. Note that if T is minimal unsolved then nothing is implied for right hand sides of constraints $T \Vdash u$ (that is, u may be a variable or not). Also, if T is minimal unsolved then all left hand sides T' with $T' \subseteq T$ are also minimal unsolved.

Lemma 2.18 *Let C be an unsolved constraint system, θ a partial solution of C, T_i a minimal unsolved left-hand side of C and u a term. If there is a simple proof of $T_i\theta \vdash u$ having the last rule an axiom or a decomposition then there is $t \in \mathrm{st}(T_i)$, t not a variable, such that $t\theta = u$.*

Proof Consider a simple proof π of $T_i\theta \vdash u$. We can suppose without loss of generality that i is minimal since if $T_j\theta \vdash u$ with $j < i$ then π' (obtained as in the definition of a simple proof) is a simple proof having as the last rule an axiom or a decomposition. We reason by induction on the depth of the proof. We can have that:

- The last rule is an axiom. Then $u \in T_i\theta$ and hence there is $t \in T_i$ (thus $t \in \mathrm{st}(T_i)$) such that $t\theta = u$. If t is a variable then $T_t \Vdash t$ is a constraint in C with $T_t \subsetneq T_i$ (see the definition of a constraint system). Hence $T_t\theta \vdash t\theta$, that is $T_t\theta \vdash u$, which contradicts the minimality of i. Thus, as required, t is not a variable.

- The last rule is a decomposition.

  Suppose that it is a symmetric decryption. That is, there is w such that $T_i\theta \vdash \{\!\{u\}\!\}_w$, $T_i\theta \vdash w$. By simplicity of the proof, the last rule applied when obtaining $\{\!\{u\}\!\}_w$ was an axiom or a decomposition, otherwise the same node would appear twice. Then applying the induction hypothesis we have that there is $t \in \mathrm{st}(T_i)$, t not a variable, such that $t\theta = \{\!\{u\}\!\}_w$.

It follows that $t = \{\!\{t'\}\!\}_{t''}$ with $t'\theta = u$. If t' is a variable then $T_{t'}\theta \vdash t'\theta$. That is $T_{t'}\theta \vdash u$, which again contradicts the minimality of i. Hence t' is not variable, as required.

For the other decomposition rules the same reasoning holds.

■

Lemma 2.19 *Let C be an unsolved constraint system, θ a partial solution and T_i a minimal unsolved left hand side of C, such that T_i does not contain two distinct non-variable subterms t_1, t_2 with $t_1\theta = t_2\theta$. If $u \in \mathrm{st}(T_i)$, u non-variable, and $T_i\theta \vdash u\theta$ then we have that $T_i' \vdash u$, where $T_i' = T_i \cup \{x \mid T \Vdash x \in C, T \subsetneq T_i\}$.*

Proof Let j be minimal such that $T_j\theta \vdash u\theta$. Thus $j \leq i$ and $T_j \subseteq T_i$. Consider a simple proof of $T_j\theta \vdash u\theta$. We reason by induction on the depth of the proof. We can have that:

- The last rule is an axiom. Then $u\theta \in T_j\theta$. If $u \in T_j$ then $T_j \vdash u$ and hence $T_i' \vdash u$. Otherwise, there is $t \in T_j$ such that $t\theta = u\theta$. We have $t \neq u$, and hence t is a variable, since otherwise there is a contradiction with the hypothesis (there are two distinct non-variable terms t and u in $\mathrm{st}(T_i)$ such that $t\theta = u\theta$). We then have $T_t\theta \vdash t\theta$. Thus $T_t\theta \vdash u\theta$, which contradicts the minimality of j, since $T_t \subsetneq T_j$. Hence $u \in T_j$ and then $T_i' \vdash u$, as required.

- The last rule is a decomposition.

  Suppose that it is the symmetric decryption rule. That is, there is w such that $T_j\theta \vdash \{\!\{u\theta\}\!\}_w$, $T_j\theta \vdash w$. The last rule applied to obtain $T_j\theta \vdash \{\!\{u\theta\}\!\}_w$ was not a composition since there are no duplicated nodes in simple proofs. We can hence apply Lemma 2.18 and obtain that there is $t \in \mathrm{st}(T_j)$, t not a variable, such that $t\theta = \{\!\{u\theta\}\!\}_w$. Since t is not a variable we have that $t = \{\!\{t'\}\!\}_{t''}$ with $t'\theta = u\theta$ and $t''\theta = w$. If t' is a variable then $T_{t'}\theta \vdash t'\theta$. Thus $T_{t'}\theta \vdash u\theta$, which contradicts the minimality of j, since $T_{t'} \subsetneq T_j$ by the definition of constraint systems. It follows that t' is not a variable. Then we have that $t' = u$ (otherwise we would have two distinct non-variable terms t' and u in $\mathrm{st}(T_i)$ with $t'\theta = u\theta$). We apply the induction hypothesis on $T_j\theta \vdash \{\!\{t'\}\!\}_{t''}\theta$ and we obtain that $T_i' \vdash \{\!\{t'\}\!\}_{t''}$. Now, if t'' is a variable then $t'' \in T_i'$, thus $T_i' \vdash t''$. Otherwise, if t'' is not a variable then, by induction hypothesis on $T_j\theta \vdash t''\theta$, we obtain $T_i' \vdash t''$. Hence, in both cases, we obtain that $T_i' \vdash t''$. Then, together with $T_i' \vdash \{\!\{t'\}\!\}_{t''}$ and $t' = u$, it follows that $T_i' \vdash u$.

  Suppose now that the last rule is the asymmetric decryption rule. That is, there is w such that $T_j\theta \vdash \{[u\theta]\}_{\mathsf{ek}(w)}$, $T_j\theta \vdash \mathsf{dk}(w)$. The last rule applied to obtain $T_j\theta \vdash \{[u\theta]\}_{\mathsf{ek}(w)}$ was not a composition. We can hence apply Lemma 2.18 and obtain that there is $t \in \mathrm{st}(T_j)$, t not a variable, such that $t\theta = \{[u\theta]\}_{\mathsf{ek}(w)}$. Since t is not a variable, and from the well-formedness of C, we have that $t = \{[t']\}_{\mathsf{ek}(w)}$ with $t'\theta = u\theta$ and w is a ground term. If t' is a variable then $T_{t'}\theta \vdash t'\theta$. Thus $T_{t'}\theta \vdash u\theta$, which contradicts the minimality of j, since $T_{t'} \subsetneq T_j$ by the definition of constraint systems. It follows that t' is not a variable. Then we have that $t' = u$ (otherwise we would have two distinct non-variable terms t' and u in $\mathrm{st}(T_i)$ with $t'\theta = u\theta$). We apply the induction hypothesis on $T_j\theta \vdash \{[t']\}_{\mathsf{ek}(w)}\theta$ and we obtain that $T_i' \vdash \{[t']\}_{\mathsf{ek}(w)}$. Applying Lemma 2.18 for $T_j\theta \vdash \mathsf{dk}(w)$, we obtain that there is $t'' \in \mathrm{st}(T_j)$, t'' not a variable such that $t''\theta = \mathsf{dk}(w)$. Using the well-formedness of C we obtain that $t'' = \mathsf{dk}(w)$. Applying now the induction hypothesis on $T_j\theta \vdash \mathsf{dk}(w)$, we get that $T_i' \vdash \mathsf{dk}(w)$. Then, together with $T_i' \vdash \{[t']\}_{\mathsf{ek}(w)}$ and $t' = u$, it follows that $T_i' \vdash u$.

  For the other decomposition rules the same reasoning as for the symmetric decryption case is applied.

- The last rule is a composition. Suppose for example that it is the symmetric encryption rule. Then $u\theta = \{\!\{w_1\}\!\}_{w_2}$ and $T_j\theta \vdash w_1$ and $T_j\theta \vdash w_2$. Since u is not a variable we have that $u = \{\!\{u_1\}\!\}_{u_2}$, $u_1\theta = w_1$ and $u_2\theta = w_2$. If u_1 (resp. u_2) is a variable then u_1 (resp. u_2) is in T_i'. Indeed, as $u_1 \in \mathrm{var}(T_i)$ (because $u \in \mathrm{st}(T_i)$), we can apply point 2 of Definition 2.1

and the minimal solvability of T_i. Otherwise (that is, if u_1 and u_2 are not variables) we apply the recursion hypothesis. Hence in both cases we have $T_i' \vdash u_1$ and $T_i' \vdash u_2$. Thus $T_i' \vdash u$.

For the other composition rules the same reasoning holds.

∎

Lemma 2.20 (completeness) *If C is an unsolved constraint system and θ is a solution of C for the property P w.r.t. L then there is a constraint system C', a substitution σ, and solution τ of C' for the property P w.r.t. $L\sigma$ such that $C \rightsquigarrow_\sigma C'$ and $\theta = \sigma\tau$.*

Proof Consider a constraint $T_i \Vdash u_i$ such that T_i is minimal unsolved and u_i is not a variable. We have $T_i\theta \vdash u_i\theta$. Consider a simple proof of $T_i\theta \vdash u_i\theta$. According to the last applied rule in this proof, we can have:

1. The last rule is a composition.

   Suppose that it is the pairing rule. That is, there are w_1, w_2 such that $T_i\theta \vdash w_1$, $T_i\theta \vdash w_2$ and $\langle w_1, w_2\rangle = u_i\theta$. Since u_i is not a variable there exists u', u'' such that $u_i = \langle u', u''\rangle$. Hence we apply the simplification rule R_{pair} in order to obtain C'. Since $u'\theta = w_1$ and $u''\theta = w_2$, the substitution θ is also a solution to C' for P w.r.t L.

   For the other composition rules the same reasoning holds, applying this time the corresponding R_f rule.

2. The last rule is an axiom or a decomposition. Applying Lemma 2.18 we obtain that there is $t \in \mathrm{st}(T_i)$, t not a variable, such that $t\theta = u_i\theta$. We have the following two possibilities:

   (a) If $t \neq u_i$ then we apply the simplification rule R_2.

   (b) Otherwise, if $t = u_i$, then $u_i \in \mathrm{st}(T_i)$. We consider the cases:

      i. There are two distinct non-variable terms $t_1, t_2 \in \mathrm{st}(T_i)$ such that $t_1\theta = t_2\theta$. Then we apply the simplification rule R_3.

      ii. Otherwise, the rule R_1 is applied. This follows from Lemma 2.19.

∎

2.2.5 Termination in polynomial time

In what follows, we first show that the new simplification rules are terminating in polynomial time. Then we show that removing already analysed constraints is a correct and complete procedure.

It is easy to show by induction on j that the following properties are satisfied.

Lemma 2.21 *Let $C = C_0$ be a constraint system, let $D_0 = \emptyset$ and $C_{i-1}; D_{i-1} \rightsquigarrow_{\sigma_i} C_i; D_i$ for all $0 < i \leq n$ for some $n > 0$. Then $C_j \cap D_j = \emptyset$ for all $0 \leq j \leq n$.*

Lemma 2.22 *Let $C = C_0$ be a constraint system, let $D_0 = \emptyset$ and $C_{i-1}; D_{i-1} \rightsquigarrow_{\sigma_i} C_i; D_i$ for all $0 < i \leq n$ for some $n > 0$. If $(T \Vdash u) \in D_n$ then there are an unique $j < n$ such that $(T \Vdash u) \in C_j \setminus C_{j+1}$ or, equivalently $(T \Vdash u) \in D_{j+1} \setminus D_j$.*

2.2.5.1 Termination

We show that each branch is of polynomial length.

Lemma 2.23 (termination) *If $C; \emptyset \rightsquigarrow_\sigma^n C'; D'$ for some constraint system C' and some substitution σ then n is polynomially bounded in the size of C.*

Proof We first notice that the rule R_4 can be applied only once. The rule R_f increases the total number of constraints by one and the rules R_2 and R_3 do not increase the total number of constraints. Thus the number of applications of rule R_1 is at most the number of constraints in C plus the number of applications of R_f. In addition, each application of R_2 or R_3 strictly decreases the number of variables. Since no rule increases the number of variables, the number of applications of the rules R_2 and R_3 is bounded by the number of variables in C. So what we need to bound is the maximum number of applications of rule R_f.

For a constraint system C'', we denote by $\mathsf{Lhs}(C'') = \{T \mid (T \Vdash u) \in C''\}$ the set of left hand sides of constraints of C''. We denote $(C_i; D_i)_{i \geq 0}$ the sequence of constraint systems obtained by applying successively the simplification rules, where $C_0 = C$ and $D_0 = \emptyset$. By Lemma 2.21, $C_k \cap D_k = \emptyset$ for all $k \geq 0$. Let j be an arbitrary index such that $(C_{j-1}; D_{j-1}) \rightsquigarrow (C_j; D_j)$ using the rule R_f. Let j_0 be the index of the last application of one of the rules R_2 or R_3, that is $j_0 < j$ and $j_0 = \max\{k < j_0 | \alpha_k \in \{R_2, R_3\}\}$ (by convention $j_0 = 0$ if there is no application of one of the rules R_2 or R_3). Suppose that, at step j, we have applied rule R_f on $T \Vdash f(u, v)$. Then $(T \Vdash f(u, v)) \in D_j$. Hence we cannot apply later (at some step $k > j$) a rule (and in particular R_f) on $T \Vdash f(u, v)$. Also note that $f(u, v) \in \mathrm{st}(C_{j_0})$. Hence until the next application of one of the rules R_2 or R_3 we can apply rule R_f at most $\sharp\mathsf{Lhs}(C_{j_0}) \times \sharp\mathrm{st}(\mathsf{rhs}(C_{j_0}))$ times, since R_2 and R_3 are the only rules that change the left-hand side of a constraint system. But $\sharp\mathsf{Lhs}(C_k) \leq \sharp\mathsf{lhs}(C_k)$ for all k (since a different left hand side T of a constraint means at least a different term). Observe now that $\sharp\mathsf{lhs}(C_i)$ (and in particular $\sharp\mathsf{lhs}(C_{j_0})$) can only decrease with regard to $\sharp\mathsf{lhs}(C_0)$ because each rule either preserves the set of terms in the left-hand side of a constraint system, or it replaces it with a new set of usually equal cardinality (but maybe smaller if some terms get unified by the application of rule R_2 or R_3). Also, $\sharp\mathrm{st}(\mathsf{rhs}(C_i)) \leq \sharp\mathrm{st}(C_i) \leq \sharp\mathrm{st}(C_0)$. Indeed, the first inclusion is trivial and the second holds because the number of subterms of C_i (w.r.t. C_{i-1}) may only decrease: this is trivial for rules R_1 and R_f, and true for R_2 and R_3 since $\sharp\mathrm{st}(u\theta) \leq \sharp(\mathrm{st}(u) \cup \mathrm{st}(v))$ when $\theta = \mathrm{mgu}(u, v)$. Hence the maximum number of applications of rule R_f is $\sharp\mathrm{var}(C_0) \times \sharp\mathsf{lhs}(C_0) \times \sharp\mathrm{st}(C_0)$.

So n is bounded by $\sharp C_0 + n_f$ (for rule R_1) plus $\sharp\mathrm{var}(C_0)$ (for rules R_2 and R_3) plus 1 (for rule R_4) plus $\sharp\mathrm{var}(C_0) \times \sharp\mathsf{lhs}(C_0) \times \sharp\mathrm{st}(C_0)$ (for rule R_f), where n_f is the number of applications of rule R_f. That is, $n \leq 1 + \sharp C + \sharp\mathrm{var}(C) + 2 \times \sharp\mathrm{var}(C) \times \sharp\mathsf{lhs}(C) \times \sharp\mathrm{st}(C)$.

Note that we also need to make sure that C' and D' are also of polynomially bounded size. This is ensured using a DAG-representation of the terms for example. ∎

2.2.5.2 Correctness

We first prove a useful lemma which states some properties of a sequence of simplification rules when rule R_1 has not been applied.

Lemma 2.24 *Let $C; \emptyset \rightsquigarrow^* C_i; D_i \rightsquigarrow^* C_n; D_n$ for some $n > i \geq 0$ be a simplification sequence such that the rule R_1 was not applied. Also let $C_i \rightsquigarrow C'_{i+1}$ using the same simplification rule as in $C_i; D_i \rightsquigarrow C_{i+1}; D_{i+1}$. If $(T \Vdash u) \in D_{i+1}$ and $\mathrm{var}(T \Vdash u) \subseteq \mathrm{var}(C'_{i+1})$ then for all j with $i < j \leq n$ such that $\mathrm{var}(T \Vdash u) \subseteq \mathrm{var}(C'_j)$ there are constraints $T \Vdash u_1, T \Vdash u_2, \ldots, T \Vdash u_k$*

in C_j and a context U that does not contain constant symbols such that $U[u_1, \ldots, u_k] = u$ and $|u_l| < |u|$ for all $1 \le l \le k$.

Proof We do the proof by induction on $|u|$. But first we derive a useful observation.

From Lemma 2.22 we know that there is $j_0 \le i$ such that $(T \Vdash u) \in (C_{j_0} \setminus C_{j_0+1})$. We prove that the rule applied at step $j_0 + 1$ cannot be R_2 or R_3. Suppose by contradiction that it is. Then, since at this step $T \Vdash u$ is removed from C_{j_0}, there is at least one variable, say x, of $T \Vdash u$ in the domain of σ_{j_0+1}. This variable does not appear in any C'_j with $j > j_0$, hence in particular it does not appear in C'_{i+1}. We obtain a contradiction. Hence at step $j_0 + 1$ the rule R_f was applied. Then $u = f(u', u'')$ and $T \Vdash u'$, $T \Vdash u''$ are in C'_{j_0+1}.

Since, as we will see later, the affirmation for $j > i + 1$ is implied by that for $j = i + 1$ we first prove this last one (i.e. for $j = i + 1$).

If $|u| = 1$ then the rule R_f couldn't have been applied (at step $j_0 + 1$) since u is atomic. Hence due to the above discussion this case is not possible.

If $|u| = 2$ then for all j with $j_0 < j \le i + 1$, $T \Vdash u'$ and $T \Vdash u''$ cannot be in D_j. Indeed otherwise (using the same argument as above for $T \Vdash u$) the rule R_f must have been applied on these constraints which would contradict that u' and u'' are atomic. Hence $T \Vdash u'$ and $T \Vdash u''$ are in C_j for all j with $j_0 < j \le i + 1$, hence in particular in C_{i+1}. Then the context is simply $U = f[\cdot, \cdot]$, $u_1 = u'$ and $u_2 = u''$.

Consider that $|u| > 2$. If $T \Vdash u'$ is in D_{i+1} then $T \Vdash u'$ is in $C_{i_0} \setminus C_{i_0+1}$ for some $i_0 \le i$ (by Lemma 2.21). Since $|u'| < |u|$ and $\mathrm{var}(T \Vdash u') \subseteq \mathrm{var}(T \Vdash u) \subseteq \mathrm{var}(C_{i+1})$ we can apply the induction hypothesis to obtain that there are constraints $T \Vdash u'_{l'}$ for $1 \le l' \le k'$ in C_{i+1} and a context U' such that $U'[u'_1, \ldots, u'_{k'}] = u'$ and $|u'_{l'}| < |u|$ for all l'. Otherwise (if $T \Vdash u'$ is in C_{i+1}) consider U' as the empty context and the set of constraints being formed by the singleton $T \Vdash u'$. The same reasoning applies for $T \Vdash u''$ obtaining the set of constraints $T \Vdash u''_{l''}$ for $1 \le l'' \le k''$ in C_{i+1} and a context U'' such that $U''[u''_1, \ldots, u''_{k''}] = u''$ and $|u''_{l''}| < |u|$ for all l''. Then take the union of these two sets of constraints and the context $f(U', U'')$ to obtain the claim.

We have found $T \Vdash u_l$ in C_{i+1} for $1 \le l \le k$ and U such that $U[u_1, \ldots, u_k] = u$ and $|u_l| < |u|$ for all l. Consider an arbitrary $j > i + 1$ such that $\mathrm{var}(T \Vdash u) \subseteq \mathrm{var}(C'_j)$. We build a new context from U by replacing the l'th hole and the corresponding constraint $T \Vdash u_l$ with new ones if $(T \Vdash u_l) \notin C_j$ and by keeping the old ones otherwise. If $(T \Vdash u_l) \notin C_j$ the new context and the corresponding constraints from C_j are obtained by applying the induction hypothesis on $T \Vdash u_l$. We can apply indeed the hypothesis since we have by Lemma 2.21 that $(T \Vdash u_l) \in D_j$, and by construction of u_l that $\mathrm{var}(T \Vdash u_l) \subseteq \mathrm{var}(C'_j)$ and $|u_l| < |u|$ (u_l is a proper subterm of u, for all l). ■

Lemma 2.25 *If $C; \emptyset \rightsquigarrow^n_\sigma C'; D'$ for some constraint system C' and some substitution σ, if θ is a solution of C' then $\sigma\theta$ is a solution of C.*

Proof Assume $C; \emptyset \rightsquigarrow^n_\sigma C'; D'$. Since applying the rule R_1 does not produce any new constraint, we can assume that the simplification rule R_1 is applied only at the end of the sequence of simplification (possibly several times). Thus $C; \emptyset \rightsquigarrow^m_{\sigma_1} C''; D'' \rightsquigarrow^p_{\sigma_2} C'; D'$ such that the rule R_1 is not applied in $C; \emptyset \rightsquigarrow^m_{\sigma_1} C''; D''$ and only the rule R_1 is applied in $C''; D'' \rightsquigarrow^p_{\sigma_2} C'; D'$. Using exactly the same argument as in the proof of Lemma 2.16 (for the case when rule R_1 was applied), we obtain that θ solution of C' implies θ solution of C''.

Thus we are reduced to proving Lemma 2.25 when the rule R_1 is not applied. Correctness now follows easily. Indeed, let τ be a solution for C_{i+1}. Using Lemma 2.24 we obtain that τ is a solution for all $(T \Vdash u) \in (C'_{i+1} \setminus C_{i+1})$ and hence τ is a solution for C'_{i+1}. Then applying Lemma 2.16 we obtained the desired result. ■

2.2.5.3 Completeness

From Lemma 2.20, we know that if C is an unsolved constraint system and θ is a solution of C for the property P w.r.t. L then there is a constraint system C' and solution τ of C' for the property P w.r.t. $L\sigma$ such that $C \rightsquigarrow_\sigma C'$ and $\theta = \sigma\tau$. Thus it is sufficient for us to show that removing already visited constraints preserves the fact that C is a constraint system.

Lemma 2.26 *If C is a constraint system and $C; \emptyset \rightsquigarrow_\sigma^* C'; D'$ then C' is a constraint system.*

Proof Let $(C_i; D_i) \rightsquigarrow_{\sigma_{i+1}} (C_{i+1}; D_{i+1})$, with $0 \leq i < n$ be the sequence of constraint systems obtained by applying successively the simplification rules, where $C_0 = C$, $D_0 = \emptyset$ and $C_n = C'$.

Again we can assume that we apply the rule R_1 last. Indeed, if there is a sequence of simplification rules leading to $C'; D'$ then there is sequence $C; \emptyset \rightsquigarrow C''; D''$ where the rule R_1 has not been applied and a sequence $C''; D'' \rightsquigarrow C'; D'$ where only rule R_1 has been applied. If C'' is a constraint system then it is easy to see (using exactly the same proof as in Lemma 2.16 for the case when rule R_1 was applied) that C' is a constraint system.

Thus we are reduced to proving the Lemma when the rule R_1 is not applied. We prove by induction on i that C_i is a constraint system. This is true for $i = 0$.

For each i we denote by C'_{i+1} the constraint system such that $C_i \rightsquigarrow_{\sigma_{i+1}} C'_{i+1}$ using the same rule as in $(C_i; D_i) \rightsquigarrow_{\sigma_{i+1}} (C_{i+1}; D_{i+1})$.

Since the simplification rules preserve constraint systems (Lemma 2.15), we have that C'_{i+1} is a constraint system. Hence the first condition of the definition of constraint systems is clearly satisfied by C_{i+1}, since it is satisfied for C'_{i+1}. We show that C_{i+1} also meets the second condition of the definition of constraint systems.

Let $(T \Vdash u) \in C_{i+1}$ and $x \in \text{var}(T)$. We have $(T \Vdash u) \in C'_{i+1}$. Hence (again by Lemma 2.15) there exists $(T_x \Vdash u_x) \in C'_{i+1}$ such that $T_x \subsetneq T$, $x \notin \text{var}(T_x)$ and $x \in \text{var}(u_x)$. If $(T_x \Vdash u_x) \in C_{i+1}$, we conclude that the second condition is satisfied. Otherwise, $(T_x \Vdash u_x) \in D_{i+1}$. Then by Lemma 2.24 we have that there are constraints $T_x \Vdash u_j$ in C_{i+1} with $1 \leq j \leq k$ for some $k > 0$ and a context U such that $U[u_1, \ldots, u_k] = u_x$. Since $x \in \text{var}(u_x)$ then there is a j with $1 \leq j \leq k$ such that $x \in \text{var}(u_j)$. Hence the constraint $T_x \Vdash u_j$ satisfies the conditions. ∎

2.2.6 An alternative approach to polynomial-time termination

In the previous section, we showed that the constraint-solving procedure finishes in polynomial-time by using a memorization technique to eliminate exponential runs. In this section we give an alternative technique using strategies.

We show next that by imposing a particular strategy on the application of simplification rules we can bound the length of every branch of the computation tree of the simplification procedure by a polynomial in the size of the initial constraint system. The strategy consists in

- first applying rules R_2 and R_3;
- next applying rules R_f respecting the descending order of the sizes of the right hand sides of constraints;
- finally apply rule R_1;
- (rule R_4, if applied, is applied last anyhow).

We first sketch the proof of completeness (corectness is independent of the order of application of rules) and next the proof of polynomial time termination.

To preserve completeness of the procedure under this strategy, we slightly relax the condition of the application of the rule R_2 on a constraint $T \Vdash u$: we require unifying a subterm $t \in \mathrm{st}(T)$ and a subterm $t' \in \mathrm{st}(u)$ (instead of unifying t with u) with $t \neq t'$, t, t' non-variables. Remark that this change preserves correctness and completeness of the initial procedure.

We have already remarked that we can "safely" apply rule R_1 last. Hence we suppose next that rules R_1 and R_4 are not applied. This leaves us with a sequence of application of rules R_2, R_3 and R_f.

Now we observe also that the application of rule R_f is "independent" of the application of rules R_2 and R_3. More precisely, we can delay arbitrarily the applications of rules R_f (with regard to the application of rules R_2 and R_3), to obtain the same result, that is the same final constraint system and the same resulting substitution (i.e. the last subscript σ). We show that we can "safely" change the application of the rule R_f followed by R_2 (each applied once) into applying first the rule R_2 and next R_f. Indeed, suppose that R_f applies first on the constraint $T \Vdash f(u_1, u_2)$ and R_2 applies next on the constraint $T' \Vdash u'$, unifying terms t and t' with $t \in \mathrm{st}(T')$, $t' \in \mathrm{st}(u')$, and $\sigma = \mathrm{mgu}(t, t')$. The resulting constraints are $T\sigma \Vdash u_1\sigma$, $T\sigma \Vdash u_2\sigma$ and $T'\sigma \Vdash u'\sigma$, possibly equal. We now (try to) apply first R_2 and next R_f. The rule R_2 is applied on $T' \Vdash u'$ if this constraint already existed in the constraint system, and on $T \Vdash f(u_1, u_2)$ otherwise, i.e. if the constraint $T' \Vdash u'$ is one of the two generated by the application of the rule R_f above. The interesting case is the latter, when $T = T'$, $u' = u_1$ (the case $u' = u_2$ is symmetric). The rule R_2 can be applied as $t' \in \mathrm{st}(f(u_1, u_2))$ (since $t' \in \mathrm{st}(u')$), and R_f can be applied on $T\sigma \Vdash f(u_1\sigma, u_2\sigma)$. We observe that the resulting substitution and set of constraints are the same in the two cases (not depending on the order of application of the two rules). We lift the above remark to arbitrary constraint systems (not only with one or two constraints). We can thus move the applications of the rule R_f in a sequence of simplification steps at the end, just as bubble-sort does. That is, we apply the above interchange first for the last application of rule R_f, making it the last step, next the last but one application of rule R_f, making it the last but one step, and so on.

The strategy requires applying rule R_f (among all the applications of rule R_f) first on the constraints with the biggest (w.r.t. the size) right hand side. In this way we are sure of not revisiting an eliminated constraint. Indeed, if a constraint $T \Vdash u$ is eliminated, at some step i, then it means the rule R_f has bean applied on it, thus $|u| = \max_{t \in \mathsf{rhs}(C_i)} |t|$. If the constraint $T \Vdash u$ is generated in C_{j+1} from C_j, for some j, then $\max_{t \in \mathsf{rhs}(C_j)} |t| > |u|$. Thus first eliminating it and then generating it (i.e. $j > i$) is not possible : since by applying rules R_f the maximum of the sizes of the right hand sides terms decreases, we have $\max_{t \in \mathsf{rhs}(C_i)} |t| \geq \max_{t \in \mathsf{rhs}(C_j)} |t|$, it follows that $|u| > |u|$. Moreover, if a constraint is eliminated during the first phase (i.e. application of rules R_2 and R_3) at step say i, then this constraint contains at least a variable which gets instantiated and hence which will not appear in the constraint systems at steps j with $j > i$.

Hence we are assured that the same constraint is never eliminated and next regenerated. Since this is the fundamental property of the approach presented in the previous section, we obtain that using this strategy the numbers of simplification steps is polynomially bounded by the size of the initial constraint systems. The proof is the same as that of Lemma 2.23 (though now we use the above argument instead of Lemmas 2.21 and 2.22 to show that visited constraints are not regenerated).

2.3 Decidability of some specialised security properties

Using the general approach presented in the previous section, verifying particular properties like the existence of key cycles or the conformation to an *a priori* given order relation on keys can be reduced to deciding these properties on solved constraint systems. We deduce a new decidability result, useful in models designed for proving cryptographic properties.

This approach also allows us to retrieve a significant fragment of [BEL04] for protocols with timestamps (in Section 2.3.2).

2.3.1 Detection of key cycles

To show that formal models (like the one presented in Chapter 1) are sound with respect to cryptographic ones, one usually assumes that no key cycle can be produced during the execution of a protocol or, even stronger, assumes that the "encrypts" relation on keys follows an *a priori* given order.

Some authors circumvent the problem of key cycles by providing new security definitions for encryption that allow key cycles [ABHS05, BPS07]. However, the standard security notions do not imply these new definitions and ad-hoc encryption schemes have to be constructed in order to satisfy the definitions. These constructions use the random oracle model which is provably non implementable. As a consequence, it is not known how to implement encryption schemes that satisfy the new definitions. In particular, none of the usual, implemented encryption schemes have been proved to satisfy the requirements.

In a passive setting, Laud [Lau02] proposed a modification of the Dolev-Yao model such that the new model is a sound abstraction even in the presence of key cycles. In his model the intruder's powers are strengthened by using a new deduction system. With the new rules, from a message containing a key cycle the intruder can infer all the involved keys. Subsequently, Janvier [Jan06] proved that the intruder deduction problem remains polynomial for the modified deduction system. It seems that this approach can be extended to active intruders and incorporated in existing tools, though to our knowledge this has not been done yet. Note that the definition of key cycles used in [Jan06] is more permissive than that of [AR02] (which is unnecessarily restrictive) and it corresponds to the approach of Laud [Lau02].

For simplicity, and since there are very few papers constraining the key relations in an asymmetric setting, in this section we restrict our attention to key cycles and key orders on symmetric keys. Moreover, we consider atomic keys for symmetric encryption since there exists no general definition (with a cryptographic interpretation) of key cycles in the case of arbitrary composed keys and soundness results are usually obtained for atomic keys. More precisely, we assume that $\mathsf{SymKey} < \mathsf{Msg}$. All function symbols of non-zero arity are of sort $\mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s}$ with $\mathsf{s} \neq \mathsf{SymKey}$. Hence only constants and variables can be of sort SymKey. In this section we call *key* a variable or a constant of sort SymKey.

2.3.1.1 Key cycles

Many definitions of key cycles are available in the literature. They are stated in terms of an "encryption" relation between keys or occurrences of keys. For example, the early definition proposed by Abadi and Rogaway [AR02], identifies a key cycle with a cycle in the encryption relation, with no conditions on the occurrences of the keys. However, the definition induced by Laud's approach [Lau02] corresponds to searching for such cycles only in the "visible" parts of a message. For example the message $\{\!\{\{\!\{k\}\!\}_k\}\!\}_{k'}$ contains a key cycle using the former definition but does not when using the latter one and assuming that k' is secret. It is generally admitted that the Abadi-Rogaway definition is unnecessarily restrictive and hence we will say that the corresponding key cycles are *strict*. However, for completeness reasons, we treat both cases.

There can still be other variants of the definition, depending whether the relation "k encrypts k'" is restricted or not to keys k' that occur in plaintext. For example, $\{\!\{\{\!\{a\}\!\}_k\}\!\}_k$ may or may not contain a key cycle. As above, even if occurrences of keys used for encrypting (as k in $\{\!\{m\}\!\}_k$) need not be considered as encrypted keys, and hence can safely be ignored when defining key cycles, we consider both cases. Note that the initial Abadi-Rogaway setting consider that $\{\!\{\{\!\{a\}\!\}_k\}\!\}_k$ has a key cycle.

We write $s <_{st} t$ if and only if s is a subterm of t. We define recursively the least reflexive and transitive relation $\sqsubseteq$ satisfying: $s_1 \sqsubseteq (s_1, s_2)$, $s_2 \sqsubseteq (s_1, s_2)$, and if $s \sqsubseteq t$ then $s \sqsubseteq \{\!\{t\}\!\}_{t'}$. Intuitively, $s \sqsubseteq t$ if s is a subterm of t that either occurs (at least once) in clear (i.e. not encrypted) or occurs (at least once) in a plaintext position. A position p is a *plaintext position* in a term u if there exists an occurrence q of an encryption in u such that $q \cdot 1 \leq p$.

Definition 2.27 *Let ρ_1 be a relation chosen in $\{<_{st}, \sqsubseteq\}$. Let S be a set of terms and k, k' be two keys. We say that k* encrypts k' in S *(denoted $k\,\rho_e^S\,k'$) if there exist $m \in S$ and a term m' such that*

$$k'\,\rho_1\,m' \text{ and } \{\!\{m'\}\!\}_k \sqsubseteq m.$$

For simplicity, we may write ρ_e instead of ρ_e^S if S is clear from the context. Also, if m is a message we denote by ρ_e^m the relation $\rho_e^{\{m\}}$.

Let S be a set of terms. We define $\mathsf{hidden}(S) \stackrel{\text{def}}{=} \{k \in \text{st}(S) \mid k \text{ of sort } \mathsf{SymKey}, S \nvdash k\}$.

Definition 2.28 (Strict key cycle) *Let K be a set of keys. We say that a set of terms S contains a* strict key cycle *on K if there is a cycle in the restriction of the relation ρ_e^S on K. Otherwise we say that S is* strictly acyclic *on K.*

We define the predicate P_{skc} as follows: P_{skc} holds on a list of terms L if and only if the set $\{m \mid L_s \vdash m\}$ contains a strict key cycle on $\mathsf{hidden}(L_s)$.

We give now the definition induced by Laud's approach [Lau02]. He has showed in a passive setting that if a protocol is secure when the intruder's power is given by a modified Dolev-Yao deduction system $\vdash_\emptyset$, then the protocol is secure in the computational model, without requiring a "no key cycle" condition. Rephrasing Laud's result in terms of the standard deduction system $\vdash$ gives rise to the below definition of key cycles, as it has been proved in [Jan06].

To state the following definition we need a more precise notion than the encrypts relation. We say that an occurrence q of a key k *is protected* by a key k' in a term m if $m|_{q''} = \{\!\{m'\}\!\}_{k'}$ for some term m' and some position q'', and the occurrence of k at q in m is a plaintext occurrence of k in m', that is $q'' \cdot 1 \leq q$. We extend this definition in the intuitive way to sets of terms. This can be done for example by indexing the terms in the set and adding this index as a prefix to the position in the term to obtain the position in the set.

Definition 2.29 (Key cycle [Jan06]) *Let K be a set of keys. We say that a set of terms S is* acyclic *on K if there exists a strict partial order $\prec$ on K such that for all $k \in K$, for all occurrences q of k in plaintext position in S there is $k' \in K$ such that $k' \prec k$ and q is protected by k' in S. Otherwise we say that S contains a* key cycle *on K.*

We define the predicate P_{kc} as follows: P_{kc} holds on a list of terms L if and only if the set $\{m \mid L_s \vdash m\}$ contains a key cycle on $\mathsf{hidden}(L_s)$.

We say that a term m contains a (strict) key cycle if the set $\{m\}$ contains one.

Exemple 2.30 *The messages $m = \{\!\!\{\{\!\!\{k\}\!\!\}_k\}\!\!\}_{k'}$ and $m' = \langle \{\!\!\{k_1\}\!\!\}_{k_2}, \{\!\!\{\{\!\!\{k_2\}\!\!\}_{k_3}\}\!\!\}_{k_1} \rangle$ are acyclic, while the message $m'' = \langle \langle \{\!\!\{k_1\}\!\!\}_{k_2}, \{\!\!\{\{\!\!\{k_2\}\!\!\}_{k_1}\}\!\!\}_{k_3} \rangle, k_3 \rangle$ has a key cycle. The orders $k' \prec k$ and $k_3 \prec k_2 \prec k_1$ prove it for m and m' while for m'' such an order cannot be found since k_3 is deducible. However, all three messages have strict key cycles.*

2.3.1.2 Key orders

In order to establish soundness of formal models in a symmetric encryption setting, the requirements on the encrypts relation can be even stronger, in particular in the case of an active intruder. In [BP04] and [JLM05] the authors require that a key never encrypts a younger key. More precisely, the encrypts relation has to be compatible with the order in which the keys are generated. Hence we also want to check whether there exist executions of the protocol for which the encrypts relation is incompatible with an *a priori* given order on keys.

Definition 2.31 (Key order) *Let $\prec$ be a strict partial order on a set of keys K. We say that a set of terms S is* compatible *with $\prec$ on K if*

$$k \, \rho_e^S \, k' \;\Rightarrow\; k' \not\preceq k, \text{ for all } k, k' \in K.$$

Given a strict partial order $\prec$ on a set of keys, we define the predicate $P_\prec$ as follows: $P_\prec$ holds on a list of terms L if and only if the set $\{m \mid L_s \vdash m\}$ is compatible with $\prec$ on $\mathsf{hidden}(L_s)$.

For example, in [BP04, JLM05] the authors choose $\prec$ to be the order in which the keys are generated: $k \prec k'$ if k has been generated before k'. We denote by $\overline{P}_\prec$ the negation of $P_\prec$. Indeed, an attack in this context is an execution such that the encrypts relation is incompatible with $\prec$, that is the predicate $\overline{P}_\prec$ holds.

The following proposition states that in the passive case a key cycle can be deduced from a set S only if it already appears in S.

Proposition 2.32 *Let L be a list of terms, and $\prec$ a strict partial order on a set of keys. The predicate P_{kc} (respectively, P_{skc} or $\overline{P}_\prec$) holds on L if and only if L_s contains a key cycle (respectively, L_s contains a strict key cycle, or the encrypts relation on L_s is not compatible with $\prec$).*

Proof The statement follows directly from the following property: if $S \vdash m$, $S \not\vdash k'$ and k' protects an occurrence q of k in m then there is an occurrence q_0 of k in S such that k' protects q_0. This property can be proved easily by induction on the depth of the proof of $S \vdash m$. The detailed proof can be found in Section 2.3.1.5. ∎

2.3.1.3 Decidability

We show how to decide the existence of key cycles or the conformation to an ordering in polynomial time for solved constraint systems. Note that the set of messages on which our predicates are applied usually contains all messages sent on the network and possibly some additional intruder knowledge.

Proposition 2.33 *Let C be a solved constraint system, L be a list of messages such that $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$, and $\prec$ a strict partial order on a set of keys. Deciding whether C has a solution for P_{kc}, P_{skc} or $\overline{P}_\prec$ w.r.t. L can be done in $\mathcal{O}(|L|^4)$.*

We devote the remaining of this section to the proof of the above proposition.

We know by Proposition 2.32 that it is sufficient to analyse the encrypts (or protects) relation only on $L_s\theta$ (and not on every deducible term), where θ is an arbitrary partial solution.

We can safely suppose that there is exactly one constraint for each variable. Indeed, eliminating from C all constraints $T' \Vdash x$ for which there is a constraint $T \Vdash x$ in C with $T \subsetneq T'$ we obtain an "equivalent" constraint system C' (that is, σ is a solution of C' iff it is a solution of C). Let t_x be the term obtained by pairing all terms of T_x (in some arbitrary order). We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. We construct the following substitution $\tau = \tau_1 \ldots \tau_n$, and τ_j is defined inductively as follows:

- $\mathrm{dom}(\tau_1) = \{x_1\}$ and $x_1\tau_1 = t_{x_1}$, i.e. $\tau_1 = \{ {}^{t_{x_1}}/_{x_1} \}$,
- $\tau_{i+1} = \tau_i \cup \{ {}^{t_{x_{i+1}}\tau_i}/_{x_{i+1}} \}$.

The construction is correct by the definition of constraint systems. It is clear that τ is a partial solution of C. We show next that it is sufficient to analyse this particular partial solution.

Key cycles. We focus first on the property P_{kc}.

Lemma 2.34 *Let C be a solved constraint system, L a list of terms such that $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$, $\mathsf{lhs}(C) \subseteq L_s$, and C has a solution for P_{kc} w.r.t L. Then τ is a solution of C for P_{kc} w.r.t L.*

We postpone its proof to Section 2.3.1.5.

Hence we just need to verify whether τ is a solution of C for P_{kc} w.r.t. L. Let $K = \mathsf{hidden}(L_s\tau)$. We build inductively the sets $K_0 = \emptyset$ and for all $i \geq 1$,

$$K_i = \{k \in K \mid \forall q \in \mathrm{pos_p}(k, L_s\tau)\, \exists k' \text{ s.t. } k' \text{ protects } q \text{ and } k' \in K_{i-1}\}$$

where $\mathrm{pos_p}(m, T)$ denotes the plaintext positions of a term m in a set T. Observe that for all $i \geq 0$, $K_i \subseteq K_{i+1}$. This can be proved easily by induction on i. Moreover, since K is finite and $K_i \subseteq K$ for all $i \geq 0$, then there is $l \geq 0$ such that $K_i = K_l$ for all $i > l$.

Lemma 2.35 *There exists $i \geq 0$ such that $K_i = K$ if and only if τ is a solution of C for P_{kc} w.r.t. L.*

Proof Consider first that there exists $i \geq 0$ such that $K_i = K$. Then take the following strict partial order on K: $k' \prec k$ if and only if there is $j \geq 0$ such that $k' \in K_j$ and $k \notin K_j$. Consider a key $k \in K$ and a plaintext occurrence q of k in $L_s\tau$. Then take $l \geq 1$ minimal such that $k \in K_l$. By the definition of K_l there is $k' \in K$ such that k' protects q and $k' \in K_{l-1}$. Since l is minimal $k \notin K_{l-1}$. Hence $k' \prec k$. Thus τ is a solution for P_{kc} w.r.t. L.

Consider now that τ is a solution. Suppose that $K_{i+1} = K_i \subsetneq K$. Let $k \in K \setminus K_{i+1}$. Since $k \notin K_{i+1}$ there is a plaintext occurrence q of k such that for all $k' \in K$ either k' does not protect q, or $k' \notin K_i$. But since τ is a solution, there is $k'' \in K$ such that k'' protects q and $k'' \prec k$. It follows that $k'' \notin K_i$, and thus $k'' \notin K_{i+1}$. Hence for an arbitrary $k \in K \setminus K_{i+1}$ we have found $k'' \in K \setminus K_{i+1}$ such that $k'' \prec k$. That is, we can build an infinite sequence $\ldots \prec k'' \prec k$ with distinct elements from a finite set – contradiction. So there exists $i \geq 0$ such that $K_i = K$. ∎

Hence to verify whether τ is a solution for P_{kc} we just need to construct the sets K_i until $K_{i+1} = K_i$ and then to test whether $K_i = K$. This algorithm is analogue to a classical method for finding a topological sorting of vertexes (and for finding cycles) of directed graphs. It is also similar to that given by Janvier [Jan06] for the intruder deduction problem considering the deduction system of Laud [Lau02].

Regarding the complexity, there are at most $\sharp K$ sets to be build and each set K_i can be constructed in $\mathcal{O}(|L_s\tau|)$. If a DAG-representation of the terms is used then $|L_s\tau| \in \mathcal{O}(|L_s|)$. This gives a complexity of $\mathcal{O}(\sharp K \times |L_s|)$ for the above algorithm.

Strict key cycles and key orders. For the other two properties P_{skc} and $\overline{P}_{\prec}$ we proceed in a similar manner. The following lemma show that it is sufficient to analyse τ when verifying the properties P_{skc} and $\overline{P}_{\prec}$.

Lemma 2.36 *Let C be a solved constraint system, L a list of terms such that* $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$ *and* $\mathsf{lhs}(C) \subseteq L_s$*, and θ a partial solution of C. For any $k, k' \in \mathsf{hidden}(L_s\theta)$, if k encrypts k' in $L_s\theta$ then k encrypts k' in $L_s\tau$.*

We give the proof of this lemma in Section 2.3.1.5 also.

We deduce that deciding whether C has a solution for P_{skc} w.r.t. L can be done simply by deciding whether the restriction of the relation $\rho_e^{L_s\tau}$ to $K \times K$ is cyclic.

Deciding whether C has a solution for $\overline{P}_{\prec}$ w.r.t. L can be done by deciding whether the restriction to $K \times K$ of the relation $\rho_e^{L_s\tau}$ has the following property P: there are $k, k' \in K$ such that $k\rho_e^{L_s\tau}k'$ and $k \preceq k'$.

Testing whether the relation $\rho_e^{L_s\tau}$ is cyclic can be done by testing for cycles in the corresponding directed graph using a classic algorithm in $\mathcal{O}(|K|^2)$. And verifying the property P can be done by analysing all pairs $(k, k') \in K \times K$ hence also in $\mathcal{O}(|K|^2)$.

Verifying any of the three properties requires a preliminary step of computing $K = \mathsf{hidden}(L_s\tau)$. Since the intruder deduction problem can be solved in $\mathcal{O}(|L_s\tau|^3)$, this gives a complexity of $\mathcal{O}(|L_s|^4)$ for computing K. $|L_s\tau| = \mathcal{O}(|L_s| \times |\tau|) = \mathcal{O}(|L_s| \times |C|) = \mathcal{O}(|L_s|^2)$ (the last equality holds since C is in solved form). And since $\sharp K \in \mathcal{O}(|L_s|)$ we obtain that the overall complexity of deciding whether a solved constrained system has a solution for any of the properties P_{kc}, P_{skc}, and $\overline{P}_{\prec}$ w.r.t. to a list L with $\mathsf{lhs}(C) \subseteq L_s$ is given by the intruder deduction problem, and hence it is $\mathcal{O}(|L_s|^4)$ (more exactly $\mathcal{O}(|L| + |L_s|^4)$ if we consider transforming the list L into the set of terms L_s).

2.3.1.4 NP-completeness

Let C be a constraint system and L a list of terms such that $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$ The NP membership of deciding whether C has a solution for P_{kc}, P_{skc} or $\overline{P}_{\prec}$ w.r.t. L follows immediately from Corollary 2.12 and Proposition 2.33.

NP-hardness is obtained by adapting the construction for NP-hardness provided in [RT03]. More precisely, we consider the reduction of the 3SAT problem to our problem. For any 3SAT boolean formula we construct a protocol such that the intruder can deduce a message containing a key cycle if and only if the formula is satisfiable. The construction is the same as in [RT03] (pages 15 and 16) except that, in the last rule, the participant responds with the term $\{\!\{k\}\!\}_k$, for some fresh key k (initially secret), instead of *Secret*. Then it is easy to see that the only way to produce a key cycle on a secret key is to play this last rule which is equivalent, using [RT03], to the satisfiability of the corresponding 3SAT formula.

2.3.1.5 Proofs of lemmas

Lemma 2.37 *Let S be a set of terms, m a term and k a key such that $S \vdash m$ and $S \not\vdash k$. Then for any plaintext occurrence q of k in m there is a plaintext occurrence q_0 in S such that if there is key k' with $S \not\vdash k'$ and which protects q_0 in S then k' protects q in m.*

Proof We reason by induction on the depth of the proof of $S \vdash m$. We can have that

- the last rule is an axiom. Hence $m \in S$. Then just take $q_0 = q$.

- the last rule is a decryption. Then $S \vdash \{\!|m|\!\}_{k''}$ and $S \vdash k''$ for some $k'' \neq k$. Take the position $q_1 = 1 \cdot q$ in $\{\!|m|\!\}_{k''}$. It is an occurrence of k. Applying the induction hypothesis we obtain an occurrence q_0 of k in S such that if there is key k' with $S \not\vdash k'$ and which protects q_0 in S then k' protects q_1 in $\{\!|m|\!\}_{k''}$. Since $S \not\vdash k'$ it follows that $k'' \neq k'$ and hence k' protects q in m.

- the last rule is a another rule. In all these cases a similar analysis as in the previous case can be done.

∎

As a corollary we obtain:

Proposition 2.38 *Let L be a list of terms, and $\prec$ a strict partial order on a set of keys. The predicate P_{kc} (respectively, P_{skc} or $\overline{P}_{\prec}$) holds on L if and only if L_s contains a key cycle (respectively, L_s contains a strict key cycle, or the encrypts relation on L_s is not compatible with $\prec$).*

Proof The right to left direction is trivial since $L_s \subseteq \{m \mid L_s \vdash m\}$.

We will prove the left to right direction only for the key cycle property, the other two properties having a similar treatment. Suppose that there is no strict partial order satisfying the conditions in Definition 2.29 for $\{m \mid L_s \vdash m\}$. In other words, for any strict partial order $\prec$ on $\mathsf{hidden}(L_s)$ there is a key k and an occurrence q of k in $\{m \mid L_s \vdash m\}$ such that for any key k', k' protects q in $\{m \mid L_s \vdash m\}$ implies $k' \not\prec k$. Using the previous lemma we can replace $\{m \mid L_s \vdash m\}$ by L_s in the previous phrase, thus obtaining that there is a key cycle in L_s. ∎

The next lemma will be used to show that $\mathsf{hidden}(L_s\theta) = \mathsf{hidden}(L_s\tau)$ for any partial solution θ.

Lemma 2.39 *Let $T \Vdash x$ be a constraint of a solved constraint system C, θ a partial solution of C and m a non-variable term. If $T\theta \vdash m$ then there is a non-variable term u with* $\mathrm{var}(u) \subseteq \mathrm{var}(T)$ *such that $T \cup \mathrm{var}(T) \vdash u$ and $m = u\theta$.*

Proof We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $(T_i \Vdash u_i) \in C$, $T_i = T$ and $u_i = x$. The lemma is proved by induction on (i, l) (considering the lexicographical order) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

- (axiom rule) $m \in T_i\theta$ or m is a public constant. If the latter holds then take $u = m$. Otherwise, there is $u \in T_i$ such that $m = u\theta$. If u is a variable then there is $j < i$ such that $T_j \Vdash u$ is a constraint of C. We have $T_j\theta \vdash u\theta$. Then by induction hypothesis there is a non-variable term u' with $\mathrm{var}(u') \subseteq \mathrm{var}(T_j)$ such that $T_j \cup \mathrm{var}(T_j) \vdash u'$ and $u\theta = u'\theta$. Hence u' satisfies the conditions.

- (decomposition rule) Suppose the rule is the decryption rule. Then the premises of the rule are $T_i\theta \vdash \{\!\{m\}\!\}_k$ and $T_i\theta \vdash k$ for some term k. By induction hypothesis there are non-variable terms u_1 and u_2 with $\mathrm{var}(u_1), \mathrm{var}(u_2) \subseteq \mathrm{var}(T_i)$ such that $T_i \cup \mathrm{var}(T_i) \vdash u_1$, $T_i \cup \mathrm{var}(T_i) \vdash u_2$, $u_1\theta = \{\!\{m\}\!\}_k$ and $u_2\theta = k$. Then $u_1 = \{\!\{u\}\!\}_{u_2'}$ with $u\theta = m$ and $u_2'\theta = k$. If u is a variable then, as in the previous case, we find an u' satisfying the conditions. Suppose u is not a variable. We still need to show that $T_i \cup \mathrm{var}(T_i) \vdash u$. If u_2' is a variable then $T_i \cup \mathrm{var}(T_i) \vdash u_2'$ since $u_2' \in \mathrm{var}(T_i)$. If u_2' is not a variable then $u_2'\theta = u_2'$ (since, as keys are atomic, u_2' is a constant), hence $u_2' = u_2$. In both cases it follows that $T_i \cup \mathrm{var}(T_i) \vdash u$. The projection rule case is simpler and is treated similarly.

- (composition rule) This case follows easily from the induction hypothesis applied on the premises.

∎

The following corollary says that any two executions allow the same set of keys to be deduced.

Corollary 2.40 *Let $T \Vdash x$ be a constraint of a solved constraint system C, and θ, θ' be two partial solutions of C. Then for any key k, $T\theta \vdash k$ if and only if $T\theta' \vdash k$.*

Proof Suppose that $T\theta \vdash k$. From the previous lemma we obtain that there is a non-variable u with $\mathrm{var}(u) \subseteq \mathrm{var}(T)$ such that $T \cup \mathrm{var}(T) \vdash u$ and $k = u\theta$. Since keys are atomic and θ is a ground substitution it follows that $u = k$. Hence $T\theta' \cup \{x\theta' \mid x \in \mathrm{var}(T)\} \vdash k$. So $T\theta' \vdash k$, since θ' is a partial solution (and thus $T\theta' \vdash x\theta'$ for all $x \in \mathrm{var}(T)$) and by using the cut elimination lemma (i.e. Lemma 2.14). ∎

We are now ready to prove Lemma 2.34.

Lemma 2.34 *Let C be a solved constraint system, L a list of terms such that $\mathrm{var}(L_s) \subseteq \mathrm{var}(C)$, $\mathsf{lhs}(C) \subseteq L_s$, and C has a solution for P_{kc} w.r.t L. Then τ is a solution of C for P_{kc} w.r.t L.*

Proof We have to prove that if there is no partial order satisfying the conditions in Definition 2.29 for the set $L_s\theta$ (according to Proposition 2.32) then there is no partial order satisfying the same conditions for $L_s\tau$. Suppose that there is a strict partial order $\prec$ which satisfies the conditions for $L_s\tau$. We prove that the same partial order does the job for $L_s\theta$.

Let $C' = C \wedge (L_s \Vdash z)$ where z is a new variable. C' is a constraint system since $\mathsf{lhs}(C) \subseteq L_s$. We write C' as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. We prove by induction on i that for all $k \in \mathsf{hidden}(L_s\theta)$, for all plaintext occurrences q of k in $T_i\theta$ there is a key $k' \in \mathsf{hidden}(L_s\theta)$ such that $k' \prec k$ and k' protects q in $T_i\theta$. It is sufficient to prove this since for $i = n$ we have $T_i = L_s$. Remark also that from Corollary 2.40 applied to $L_s \Vdash z$ we obtain that $\mathsf{hidden}(L_s\theta) = \mathsf{hidden}(L_s\tau)$.

For $i = 1$ we have $T_1 = T_1\theta = T_1\tau$ hence the property is clearly satisfied for θ since it is satisfied for τ.

Let $i > 1$. Consider an occurrence q of a key $k \in \mathsf{hidden}(L_s\theta)$ in a plaintext position of w for some $w \in T_i\theta$. Let $t \in T_i$ such that $w = t\theta$.

If q is a non-variable position in t then it is a position in $t\tau$. And since τ is a solution we have that there is a key $k' \in \mathsf{hidden}(L_s\tau)$ (hence $k' \in \mathsf{hidden}(L_s\theta)$) such that $k' \prec k$ and q is protected by k' in $t\tau$. The key k' cannot occur in some $x\tau$, with $x \in \mathrm{var}(t)$ since otherwise k' is deducible (indeed $x\tau = k'$ since the keys are atomic and $T_x\tau \vdash x\tau$). Hence k' occurs in t. Then k' protects q in t, and thus in w also.

If q is not a non-variable position in t then there is a variable $x_j \in \mathrm{var}(t)$ with $j < i$ such that the occurrence q in $t\theta$ is an occurrence of k in $x_j\theta$ (formally $q = p \cdot q'$ where p is some position of x_j in t and q' is some occurrence of k in $x_j\theta$). Applying Lemma 2.37 we obtain that there is an occurrence q_0 of k in $T_j\theta$ such that if there is a key k' with $T_j\theta \not\vdash k'$ and which protects q_0 in $T_j\theta$ then k' protects q' in $x_j\theta$. The existence of the key k' is assured by the induction hypothesis on $T_j\theta$. Hence k' protects q' in $x_j\theta$ and thus q in w. ∎

Lemma 2.41 *Let $T \Vdash x$ be a constraint of a solved constraint system C and θ be a partial solution. Let m, u, k be terms such that*

$$T\theta \vdash m \text{ and } \{\!\!\{u\}\!\!\}_k \sqsubseteq m \text{ and } T\theta \not\vdash k.$$

Then there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T$ and $v\theta = \{\!\!\{u\}\!\!\}_k$.

Proof We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $(T_i \Vdash x_i) \in C$, $T_i = T$ and $x_i = x$. The lemma is proved by induction on (i, l) (lexicographical order) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

- (axiom rule) $m = t\theta$ for some $t \in T_i$. We can have that either there is $t' \sqsubseteq t$ such that $t'\theta = \{\!\{u\}\!\}_k$, or $\{\!\{u\}\!\}_k \sqsubseteq y\theta$ for some $y \in \text{var}(t)$. In the first case take $v = t'$, $w = t$. In the second case, by the definition of constraint systems, there exists $(T_j \Vdash y) \in C$ with $j < i$. Since $T_j\theta \vdash y\theta$ and $T_j\theta \nvdash k$ (since $T_j \subseteq T_i$), we deduce by induction hypothesis that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_j$, hence $w \in T_i$ and $v\theta = \{\!\{u\}\!\}_k$.

- (decomposition rule) Let m' be the premise of the rule. We have that $T_i\theta \vdash m'$ (with a proof of a strictly smaller length) and $m \sqsubseteq m'$ thus $\{\!\{u\}\!\}_k \sqsubseteq m'$. By induction hypothesis, we deduce that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_i$, and $v\theta = \{\!\{u\}\!\}_k$.

- (composition rule) All cases are similar to the previous one except if $m = \{\!\{u\}\!\}_k$ and the rule is $\dfrac{S \vdash x \quad S \vdash y}{S \vdash \{\!\{x\}\!\}_y}$. But this case contradicts $T_i\theta \nvdash k$.

■

The following simple lemma is also needed for the proof of Lemma 2.36.

Lemma 2.42 *Let $T \Vdash x$ be a constraint of a solved constraint system C, θ be a partial solution, $k \in \mathsf{hidden}(T\theta)$, and m a term such that $T\theta \vdash m$. If $k\,\rho_1\,m$ then there is $t \in T$ such that $k\,\rho_1\,t$.*

Proof We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $(T_i \Vdash u_i) \in C$, $T_i = T$ and $u_i = x$. The lemma is proved by induction on (i, l) (considering the lexicographical order) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

- (axiom rule) $m \in T_i\theta$ or m a public constant. If m is a public constant then $k \neq m$ since $k \in \mathsf{hidden}(T\theta)$. Thus there is $t \in T_i$ such that $m = t\theta$. If $k\,\rho_1\,t$ then we're done. Otherwise there is a variable $y \in \text{var}(t)$ such that $k\,\rho_1\,y\theta$. Also, there is $j < i$ such that $T_j \Vdash y$ is a constraint of C. Then, by induction hypothesis, there is $t' \in T_j$, hence in T_i, such that $k\,\rho_1\,t'$.

- (composition or decomposition rule) By inspection of all the composition and decomposition rules we observe that there is always a premise $T_i\theta \vdash m'$ with $k\,\rho_1\,m'$ for some term m'. The conclusion follows then directly from the induction hypothesis.

■

We can prove now Lemma 2.36.

Lemma 2.36 *Let C be a solved constraint system, L a list of terms such that $\text{var}(L_s) \subseteq \text{var}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$, and θ a partial solution of C. For any $k, k' \in \mathsf{hidden}(L_s\theta)$, if k encrypts k' in $L_s\theta$ then k encrypts k' in $L_s\tau$.*

Proof Remember that $\mathsf{hidden}(L_s\theta) = \mathsf{hidden}(L_s\tau)$ (from Corollary 2.40, as shown in the proof of Lemma 2.34).

Consider two keys $k, k' \in \mathsf{hidden}(L_s\theta)$ such that k encrypts k' in $L_s\theta$. Then there are terms u, u' such that $u' \in L_s\theta$, $\{\!\{u\}\!\}_k \sqsubseteq u'$ and $k'\,\rho_1\,u$. We can have that either (first case) there are v, w such that $v \sqsubseteq w \in L_s$, v non-variable and $\{\!\{u\}\!\}_k = v\theta$, or (second case) $\{\!\{u\}\!\}_k \sqsubseteq x\theta$ with $x \in \text{var}(L_s)$. In the second case, consider the constraint $(T_x \Vdash x) \in C$. We have $T_x\theta \vdash x\theta$. Hence we can apply Lemma 2.41 for $x\theta$, u and k to obtain that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_x$ and $v\theta = \{\!\{u\}\!\}_k$. Hence, in both cases, we obtained that there is a non-variable term $v \in \mathrm{st}(L_s)$ (since $T_x \subseteq L_s$) such that $v\theta = \{\!\{u\}\!\}_k$. Thus there is v_0 such that $v = \{\!\{v_0\}\!\}_k$. Indeed, otherwise $v = \{\!\{v_0\}\!\}_y$ for some $y \in \mathrm{var}(L_s)$, hence $y \in \mathrm{var}(C)$. Since C is solved we have $T_y\sigma \vdash y\sigma$. But $y\sigma = k$, contradicting $k \in \mathsf{hidden}(L_s\theta)$.

We have $v_0\theta = u$. Since $k' \,\rho_1\, u$ and k' is a constant or a variable, we can have that $k' \,\rho_1\, v_0$, or $k' \,\rho_1\, y\theta$ for some $y \in \mathrm{var}(v_0)$. If $k' \,\rho_1\, v_0$ then k encrypts k' in L_s, hence in $L_s\tau$ also. If $k' \,\rho_1\, y\theta$ then from the previous lemma $k' \,\rho_1\, t$ for some $t \in T_y$, and hence $k' \,\rho_1\, y\tau$. Therefore in both cases we have that k encrypts k' in $L_s\tau$. ∎

2.3.2 Secrecy for protocols with timestamps

For modeling timestamps, we introduce a new sort $\mathsf{Time} < \mathsf{Msg}$ for time and we assume an infinite number of constants of sort Time, represented by rational numbers or integers[11]. We assume that the only two sorts are Time and Msg. Any value of time should be known to an intruder, that is why we add to the deduction system the rule $\dfrac{}{S \vdash x}$ where x is a variable of sort Time. All the previous results can be easily extended to such a deduction system since ground deducibility remains decidable in polynomial time.

To express relations between timestamps, we use timed constraints. An *integer timed constraint* or a *rational timed constraint* T is a conjunction of formulas of the form

$$\Sigma_{i=1}^{k} \alpha_i x_i \ltimes \beta,$$

where the α_i and β are rational numbers, $\ltimes \in \{<, \leq\}$, and the x_i are variables of sort Time. A *solution* of a rational (resp. integer) timed constraint T is a closed substitution $\sigma = \{ {}^{c_1}/_{x_1}, \ldots, {}^{c_k}/_{x_k} \}$, where the c_i are rationals (resp. integers), that satisfies the constraint.

Timed constraints between the variables of sort Time are expressed through satisfiability of security properties.

Definition 2.43 *A predicate P is a* timed property *if P is* generated *by some (rational or integer) timed constraint T, that is if T has variables $x_1, \ldots, x_k$ then for any list L of messages $P(L)$ holds if and only if*

- *L contains exactly k messages $t_1, \ldots, t_k$ of sort* Time *that appear in this order in the list, and*
- *$T(t_1, \ldots, t_k)$ is true.*

Such timed properties can be used for example to say that a timestamp x_1 must be fresher than a timestamp x_2 ($x_1 \geq x_2$) or that x_1 must be at least 30 seconds fresher than x_2 ($x_1 \geq x_2 + 30$).

Exemple 2.44 *We consider the Wide Mouthed Frog Protocol [CJ97].*

$$\begin{array}{rl} A \to S : & A, \{\!\{T_a, B, K_{ab}\}\!\}_{K_{as}} \\ S \to B : & \{\!\{T_s, A, K_{ab}\}\!\}_{K_{bs}} \end{array}$$

A sends to a server S a fresh key K_{ab} intended for B. If the timestamp T_a is fresh enough, the server answers by forwarding the key to B, adding its own timestamps. B simply checks whether

[11]This can be achieved formally by considering only one constant 0 of sort Time, and a function symbol succ of sort $\mathsf{Time} \to \mathsf{Time}$. For simplicity, we omit these technicalities.

this timestamp is older than any other message he has received from S. As explained in [CJ97], this protocol is flawed because an attacker can use the server to keep a session alive as long as he wants by replaying the answers of the server.

This protocol can be modeled by the following constraint system:

$$S_1 \stackrel{\text{def}}{=} \{a, b, s, \langle a, \{\!\{0, b, k_{ab}\}\!\}_{k_{as}}\rangle\} \quad \Vdash \quad \langle a, \{\!\{x_{t_1}, b, y_1\}\!\}_{k_{as}}\rangle, x_{t_2} \tag{2.4}$$

$$S_2 \stackrel{\text{def}}{=} S_1, \{\!\{x_{t_2}, a, y_1\}\!\}_{k_{bs}} \quad \Vdash \quad \langle b, \{\!\{x_{t_3}, a, y_2\}\!\}_{k_{bs}}\rangle, x_{t_4} \tag{2.5}$$

$$S_3 \stackrel{\text{def}}{=} S_2, \{\!\{x_{t_4}, b, y_2\}\!\}_{k_{as}} \quad \Vdash \quad \langle a, \{\!\{x_{t_5}, b, y_3\}\!\}_{k_{as}}\rangle, x_{t_6} \tag{2.6}$$

$$S_4 \stackrel{\text{def}}{=} S_3, \{\!\{x_{t_6}, a, y_3\}\!\}_{k_{bs}} \quad \Vdash \quad \{\!\{x_{t_7}, a, k_{ab}\}\!\}_{k_{bs}} \tag{2.7}$$

where y_1, y_2, y_3 are variables of sort Msg *and $x_{t_1}, \dots, x_{t_7}$ are variables of sort* Time*. We add explicitly the timestamps emitted by the agents on the right hand side of the constraints (that is in the messages expected by the participants) since the intruder can schedule the message transmission whenever he wants.*

Initially, the intruder simply knows the identities of the agents and A's message at time 0. Then S answers alternatively to requests from A and B. Since the intruder controls the network, the messages can be scheduled as slow (or fast) as the intruder needs it. The server S should not answer if A's timestamp is too old (let's say older than 30 seconds) thus S's timestamp cannot be too much delayed (no more than 30 seconds). This means that we should have $x_{t_2} \leq x_{t_1} + 30$. Similarly, we should have $x_{t_4} \leq x_{t_3} + 30$ and $x_{t_6} \leq x_{t_5} + 30$. The last rule corresponds to B's reception. In this scenario, B does not perform any check on the timestamp since it is the first message he receives.

We say that there is an attack if there is a solution to the constraint system that satisfies the previously mentioned time constraints and such that the timestamp received by B is too fresh to come from A: $x_{t_7} \geq 30$. Formally, we consider the timed property generated by the following timed constraint:

$$x_{t_2} \leq x_{t_1} + 30 \;\wedge\; x_{t_4} \leq x_{t_3} + 30 \;\wedge\; x_{t_6} \leq x_{t_5} + 30 \wedge\; x_{t_7} \geq 30.$$

Then the substitution corresponding to the attack is

$$\sigma = \{{}^{k_{ab}}/_{y_1}, {}^{k_{ab}}/_{y_2}, {}^{k_{ab}}/_{y_3}, {}^{k_{ab}}/_{y_4}, {}^{0}/_{x_{t_1}}, {}^{30}/_{x_{t_2}}, {}^{30}/_{x_{t_3}}, {}^{60}/_{x_{t_4}}, {}^{60}/_{x_{t_5}}, {}^{90}/_{x_{t_6}}, {}^{90}/_{x_{t_7}}\}.$$

Proposition 2.45 *Any timed property can be decided in non-deterministic polynomial time on solved constraint systems.*

Proof Let C be a solved constraint system, P a timed property and T a timed constraint generating P. Let $y_1, \dots, y_n$ be the variables of sort Msg in C and $x_1, \dots, x_k$ the variables of sort Time in C. Clearly, any substitution σ of the form $y_i\sigma = u_i$ where $u_i \in S_i$ for some $(S_i \Vdash y_i) \in C$ and $x_i\sigma = t_i$ for t_i any constant of sort Time is a solution of C for the **true** property. Let σ' be the restriction of σ to the timed variables $x_1, \dots, x_k$.

Clearly, σ is a solution of C for P if and only if σ' is a solution to T. Thus there exists a solution of C for P if and only if T is satisfiable. The satisfiability of T is solved by usual linear programming [Sch98]. It is polynomial in the case of rational timed constraints and it is NP-complete in the case of integer timed constraints, thus the result. ∎

2.3.2.1 NP-completeness

We deduce by combining Theorem 2.10 and Proposition 2.45 that the problem of deciding timed properties on arbitrary constraint systems is in NP.

NP-hardness directly follows from the NP-hardness of constraint system solving by considering a predicate corresponding to an empty timed constraint.

2.4 Conclusions

We have shown how the generic approach we have derived from [CLS03, RT03] can be used to retrieve two NP-completeness results. The first one allows us to detect key cycles, and the second one to solve constraint systems with timed constraints. In the two cases, we had to provide a decision procedure only for a simple class of constraint systems. Since the constraint-based approach [CLS03, RT03] has already been implemented in Avispa [ABB$^+$05], we plan, using our results, to adapt this implementation to the case of key cycles and timestamps.

Regarding key cycles, our approach is valid for a bounded number of sessions only. Secrecy is undecidable in general [DLM04] for an unbounded number of sessions. Such an undecidability result could be easily adapted to the problem of detecting key cycles. Several decidable fragments have been designed [RS03, CLC03a, BP03b, VSS05] for secrecy and an unbounded number of sessions. We plan to investigate how such fragments could be used to decide key cycles.

Chapter 3

Decidability results for Horn clauses. Application to protocols using CBC encryption and blind signatures

Recently, several procedures for deciding secrecy have been proposed for operators with algebraic properties for a bounded [CKRT03, DLLT06, BC06, DLL07] or unbounded [CLC03a, VSS05] number of sessions. The properties of blind signatures were considered in [BC06], and in [CKRT03] the prefix property (which is very similar to homomorphism) is also handled along with the properties of XOR. Homomorphism theory with associative-commutative operators is considered in [LLT05] for the case of a passive intruder, and homomorphism with XOR or abelian groups is considered for an active intruder in [DLLT06, Del06]. An electronic voting protocol has been analysed in [KR05]. The protocol relies on a blind signature scheme whose properties have been modeled by equations; secrecy of votes have been proved automatically using the ProVerif tool by B. Blanchet [Bla01]. This tool can handle an unbounded number of sessions and arbitrary equational theories [BAF05] but it does not guarantee termination (even in the absence of equations).

The above mentioned works do not address the decidability of secrecy with CBC encryption or blind signatures in the case of an unbounded number of sessions. In this chapter, we consider exactly this setting. Following the line of [CLC03a], we tackle the problem by introducing a new fragment of Horn clauses. We show the decidability of this fragment using a combination of several resolution strategies. We apply this result to fix the Needham-Schroeder symmetric key authentication protocol, which is known to be flawed when CBC mode is used.

Outline of the chapter In Section 3.1, we introduce Horn clauses and explain how protocols can be modeled using them. We then introduce the new fragment of first order clauses in Section 3.2. In Section 3.3, we present our resolution strategy and apply it to this fragment, proving that this strategy is both complete and terminating for this class. The application to the Needham-Schroeder symmetric key protocol is shown in Section 3.4.

3.1 The model

The aim of this section is to introduce Horn clauses and show how we use them to model cryptographic protocols.

3.1.1 Horn clauses

To our purposes a single unary predicate suffices. Let I be this predicate. *Atoms* A are of the form $I(u)$ where u is a term. *Literals* L are either positive literals $+A$ (or simply A) or negative literals $-A$ where A is an atom. A *clause* is a finite set of literals. If C_1 and C_2 are clauses, $C_1 \vee C_2$ denotes $C_1 \cup C_2$. A *Horn clause* is a clause that contains at most one positive literal. For Horn clauses we may use the alternative notation $A_1, A_2, \ldots, A_{n-1} \rightarrow A_n$ to denote the clause $-A_1 \vee -A_2 \vee \ldots \vee -A_{n-1} \vee A_n$. We distinguish from the context the atom $I(u)$ and the clause $I(u)$ consisting of the positive literal $+I(u)$.

If M is an atom, a literal, a clause, or a set of such objects, and σ is a substitution, then $M\sigma$ obtained by applying σ to M is defined as usual. For example, for an atom $A = I(u)$ and a substitution σ, $A\sigma$ denotes the atom $I(u\sigma)$. We also extend as usual the notation of unifier from terms to literals.

A *(Herbrand) interpretation* is a set of ground atoms. An ground atom $I(u)$ is *true* in the interpretation $\mathfrak{I}$ if $u \in \mathfrak{I}$, and it is *false* otherwise. A ground clause C is *satisfied* by $\mathfrak{I}$ if and only if there is a positive literal $+I(u) \in C$ such that $I(u)$ is true in $\mathfrak{I}$ or there is a negative literal $-I(u) \in C$ such that $I(u)$ is false in $\mathfrak{I}$. A clause C is satisfied by $\mathfrak{I}$ if for all ground substitutions σ, $C\sigma$ is satisfied by $\mathfrak{I}$. If C is satisfied by $\mathfrak{I}$ then we say that $\mathfrak{I}$ is a *model* of C; we say that $\mathfrak{I}$ is a model of a set of clauses $\mathcal{C}$ if it is a model of all clauses in $\mathcal{C}$. A clause (or a set of clauses) is *satisfiable* if it has a model, and *unsatisfiable* otherwise.

Given two sets of clauses $\mathcal{C}$ and $\mathcal{C}'$, we say that $\mathcal{C}'$ is a *logical consequence* of $\mathcal{C}$ if every model of $\mathcal{C}$ is also a model of $\mathcal{C}'$. When $\mathcal{C}'$ is a singleton consisting of C' then we simply write $\mathcal{C} \models C'$ instead of $\mathcal{C} \models \{C'\}$. It is easy to see that, for a ground term m, $\mathcal{C} \cup \{-I(m)\}$ is unsatisfiable if and only if $\mathcal{C} \models I(m)$.

If u is a term, $\|u\|$ is the *depth* of u, that is 1 plus the maximal length of the positions of u. For a variable x, $\|u\|_x$ is the maximal depth of x in u, that is 1 plus the maximal length of the occurrences of x in u. By convention, if x is a variable and $x \notin \mathrm{var}(u)$ then $\|u\|_x = 0$. The definitions of $\|\cdot\|$ and $\|\cdot\|_x$ are extended to literals by $\|\pm I(u)\| = \|u\|$ and $\|\pm I(u)\|_x = \|u\|_x$.

3.1.1.1 An ordering on terms

We consider a strict and total ordering $<_{\mathcal{F}}$ on the function symbols. We define next a partial ordering on terms. The ordering is chosen in order to ensure the termination of our resolution procedure.

Definition 3.1 (ordering $<$) *Let u and v be two terms. We say that $u < v$ if one of following two conditions holds:*

1. $\|u\| < \|v\|$, *and* $\|u\|_x < \|v\|_x$ *for every* $x \in \mathrm{var}(u) \cup \mathrm{var}(v)$*;*
2. $\|u\| \leq \|v\|$, $\|u\|_x \leq \|v\|_x$ *for every* $x \in \mathrm{var}(u) \cup \mathrm{var}(v)$*, u and v are not variables or names, and one of the following two conditions holds:*
   - *(a)* $\mathrm{head}(u) <_{\mathcal{F}} \mathrm{head}(v)$,
   - *(b)* $\mathrm{head}(u) = \mathrm{head}(v)$, $\forall i, u_i \leq v_i$, *and* $\exists i$ *such that* $u_i < v_i$*, where* $u = f(u_1, \ldots, u_n)$ *and* $v = f(v_1, \ldots, v_n)$*, for some* $f \in \mathcal{F}$ *of arity* $n \geq 0$.

For example, if u is a strict subterm of v then $u < v$. Variables (and names) are incomparable. We have $\langle a, x\rangle < \mathsf{h}(\mathsf{h}(x))$ but $\langle \mathsf{h}(\mathsf{h}(a)), x\rangle \not< \mathsf{h}(\mathsf{h}(x))$. We also have that $\{\!\{x\}\!\}_z < \{\!\{\langle x, y\rangle\}\!\}_z$.

An ordering is said *liftable* if for any two terms u, v and for any substitution θ, $u < v$ implies $u\theta < v\theta$. This is a crucial property for the completeness of ordered resolution.

Proposition 3.2 *The relation $<$ is a strict liftable ordering.*

Proof Transitivity and irreflexivity of $<$ are obvious. We have

$$\|w\sigma\| = \max(\|w\|, \max_{y \in \mathrm{var}(w)} (\|w\|_y + \|y\sigma\| - 1)) \tag{3.1}$$

and

$$\|w\sigma\|_x = \begin{cases} 0 & \text{if } x \notin \mathrm{var}(w\sigma), \\ \max_{y \in \mathrm{var}(w)} (\|w\|_y + \|y\sigma\|_x - 1) & \text{otherwise.} \end{cases}$$

Also, $\mathrm{head}(w\sigma) = \mathrm{head}(w)$ if w is not a variable. Moreover, if $u < v$ then $\mathrm{var}(u) \subseteq \mathrm{var}(v)$, since otherwise there exists $x \in \mathrm{var}(u) \setminus \mathrm{var}(v)$ with $\|u\|_x > \|v\|_x = 0$ (contradiction). Remark that $\mathrm{var}(w\sigma) = \cup_{x \in \mathrm{var}(w)} \mathrm{var}(x\sigma)$. Thus, if $x \in \mathrm{var}(u\sigma)$ and $\mathrm{var}(u) \subseteq \mathrm{var}(v)$ then $x \in \mathrm{var}(v\sigma)$.

Let u and v be terms such that $u < v$. We prove by induction on the depth of u that $u\sigma < v\sigma$.

For the base case, $\|u\| = 1$ and then u is either a variable, a name or a constant. If $\|v\| = 1$ then u and v are constants, and thus so are $u\sigma$ and $v\sigma$. Suppose $\|v\| > 1$. If u is a name or a constant then $u\sigma < v\sigma$ by using the first point of Definition 3.1. If u is a variable x then $x \in \mathrm{var}(v)$ and thus $u\sigma = x\sigma < v\sigma$ (since $x\sigma$ is a subterm of $v\sigma$).

We consider now the inductive case ($\|u\| > 1$).

Suppose $\|u\| < \|v\|$, and $\|u\|_x < \|v\|_x$ for every $x \in \mathrm{var}(u) \cup \mathrm{var}(v)$. Then for all $x \in \mathrm{var}(u\sigma) \cup \mathrm{var}(v\sigma)$, $\|u\sigma\|_x = \max_{y \in \mathrm{var}(u)}(\|u\|_y + \|y\sigma\|_x - 1) < \max_{y \in \mathrm{var}(u)}(\|v\|_y + \|y\sigma\|_x - 1) \leq \max_{y \in \mathrm{var}(v)}(\|v\|_y + \|y\sigma\|_x - 1) = \|v\sigma\|_x$. Next, using the just obtained strict inequality, equation 3.1, and that $\mathrm{var}(u) \subseteq \mathrm{var}(v)$ we obtain that $\|u\sigma\| < \|v\sigma\|$. Thus, by the first point of Definition 3.1, $u\sigma < v\sigma$.

Suppose now that $\|u\| \leq \|v\|$, and $\|u\|_x \leq \|v\|_x$ for every $x \in \mathrm{var}(u) \cup \mathrm{var}(v)$. Then using a similar analysis as in the previous case we obtain that $\|u\sigma\| \leq \|v\sigma\|$, and $\|u\sigma\|_x \leq \|v\sigma\|_x$ for every $x \in \mathrm{var}(u\sigma) \cup \mathrm{var}(v\sigma)$. If $\mathrm{head}(u) <_{\mathcal{F}} \mathrm{head}(v)$ then $\mathrm{head}(u\sigma) <_{\mathcal{F}} \mathrm{head}(v\sigma)$, thus $u\sigma < v\sigma$. Otherwise (i.e. $\mathrm{head}(u) = \mathrm{head}(v)$), we have and $\forall i, u_i \leq v_i$, and $\exists i$ such that $u_i < v_i$ where $u = f(u_1, \ldots, u_n)$ and $v = f(v_1, \ldots, v_n)$ for some $f \in \mathcal{F}$, $n > 0$. Applying the induction hypothesis we obtain that $\forall i, u_i\sigma \leq v_i\sigma$, and $\exists i$ such that $u_i\sigma < v_i\sigma$. Then clearly $u\sigma < v\sigma$ (by point 2b of Definition 3.1). ■

A term v is said *maximal* in a set S if there is no term $u \in S$ such that $v < u$.

The ordering is extended to literals by $\pm I(u) < \pm' I(v)$ if and only if $u < v$.

3.1.2 From protocols to Horn clauses

The predicate I represents the knowledge of the intruder: $I(m)$ means that the intruder knows the term (or message) m. Thus a clause $I(u_1), \ldots, I(u_n) \to I(v)$ should be read as "if the intruder knows some messages of the form $u_1, \ldots, u_n$ respectively, then he knows v". There is thus a natural correspondence between deductions and Horn clauses, as we will see in what follows.

3.1.2.1 Intruder clauses

Let $\mathcal{I}$ be a deduction system without conditions. The set of clauses associated with $\mathcal{I}$ is

$$\mathcal{C}_{\mathcal{I}} \stackrel{\text{def}}{=} \{I(u_1), \ldots, I(u_n) \to I(v) \mid \frac{S \vdash u_1 \quad \ldots \quad S \vdash u_n}{S \vdash v} \in \mathcal{I}\}.$$

Exemple 3.3 *The Dolev-Yao rules presented in Section 1.2.3.1 (page 41), more exactly the rules of $\mathcal{I}_d$ from Figure 1.3 and the composition rules (for the function symbols occurring in Figure 1.3), are represented by the following set of Horn clauses:*

$$
\begin{array}{ll}
I(x), I(y) \to I(\langle x, y\rangle) & \textit{pairing of messages} \\
I(x), I(y) \to I(\{\!\!\{x\}\!\!\}_y) & \textit{symmetric encryption} \\
I(x), I(y) \to I(\{\![x]\!\}_y) & \textit{asymmetric encryption} \\
I(x), I(y) \to I([\![x]\!]_y) & \textit{digital signing} \\
I(x) \to I(\mathsf{ek}(x)) & \textit{obtaining the encryption key} \\
I(\langle x, y\rangle) \to I(x) & \textit{first projection} \\
I(\langle x, y\rangle) \to I(y) & \textit{second projection} \\
I(\{\!\!\{x\}\!\!\}_y), I(y) \to I(x) & \textit{symmetric decryption} \\
I(\{\![x]\!\}_{\mathsf{ek}(y)}), I(\mathsf{dk}(y)) \to I(x) & \textit{asymmetric decryption}
\end{array}
$$

Observe that the deduction system $\mathcal{I}(\mathcal{R}_0)$ (see Definition 1.8 and Figure 1.2) contains also the membership rule and an infinite number of composition rules for public constants (hence the set $\mathcal{C}_{\mathcal{I}(\mathcal{R}_0)}$ is also infinite). The membership rule is a conditional rule and is hence not represented by an associated clause. As it we will see next, we represent explicitly the initial knowledge of some set of terms (i.e. the left hand side S of a deduction rule $S \vdash u$).

Given a set of terms T, we define the associated set of clauses $\mathcal{C}_T \stackrel{\text{def}}{=} \{I(u) \mid u \in T\}$.

Lemma 3.4 *Let $\mathcal{I}$ be a deduction system without conditions, T a set of ground terms and m a ground term. If $T \vdash_{\mathcal{I}} m$ then $\mathcal{C}_{\mathcal{I}} \cup \mathcal{C}_T \models I(m)$.*

Proof Consider a proof of $T \vdash m$. We reason by induction on the depth of the proof. Then, by considering the last rule of the proof we can have the following possibilities.

- The last rule is an axiom and $m \in T$. Then $I(m) \in \mathcal{C}_T$ and hence $\mathcal{C}_T \models I(m)$.
- The last rule is
$$\frac{S \vdash u_1 \quad \ldots \quad S \vdash u_n}{S \vdash v}$$
and $v\sigma = m$ for some substitution σ. Then $(I(u_1), \ldots, I(u_n) \to I(v)) \in \mathcal{C}_{\mathcal{R}}$, and, by induction hypothesis, $\mathcal{C}_{\mathcal{R}} \cup \mathcal{C}_T \models I(u_i\sigma)$ for all i. Consider an arbitrary model $\mathfrak{I}$ of $\mathcal{C}_{\mathcal{R}} \cup \mathcal{C}_T$. Then $\mathfrak{I}$ is also a model for all $I(u_i\sigma)$. If $I(v\sigma)$ is false in $\mathfrak{I}$ then the clause $I(u_1\sigma), \ldots, I(u_n\sigma) \to I(v\sigma)$ is not satisfied by $\mathfrak{I}$. Hence neither is the clause $I(u_1), \ldots, I(u_n) \to I(v)$. But this contradicts $\mathfrak{I}$ being a model of $\mathcal{C}_{\mathcal{R}} \cup \mathcal{C}_T$. Thus $I(v\sigma)$ is true in $\mathfrak{I}$. That is, $\mathfrak{I}$ is also a model of $I(m)$.

■

3.1.2.2 Protocol clauses

We now show how the rules of a protocol can also be modeled using Horn clauses. With every rule we associate a clause. We perform first some abstraction by letting the nonces only depend on the agent that has created it and the agent that should receive it; and similarly, a fresh session key will be parameterised by the agents who share the key. This is formalised next.

We consider again only roles with matching and simply call them roles. Let $\Pi = (\mathcal{R}, \mathcal{S})$ be a k-party protocol. We consider the partial function $\mathcal{D}$ given by $\mathcal{D}(r, p) = (r', p')$ if and only if

$\mathcal{S}(r',p') = (r,p)$. That is, $\mathcal{D}$ returns for each role/control-point pair (r,p), the role/control-point pair (r',p') of the expected destination of the message sent by role r at step p. The function $\mathcal{D}$ is used to obtain, for a fresh nonce or key, which is the agent that should receive it.

Let $R_r(z_1,\ldots,z_k) = \nu\widetilde{x}_r.\ \mathsf{recv}(u_r^1), \mathsf{send}(v_r^1);\ \ldots\ ;\ \mathsf{recv}(u_r^{p_r}), \mathsf{send}(v_r^{p_r})$ with $1 \leq r \leq k$ be the roles of Π. We assume that no different roles have common variables (hence no common fresh items neither), and we let $\cup_{1\leq r\leq k}\widetilde{x}_r = \{x_1,\ldots,x_n\}$. We also suppose that the signature $\mathcal{F}$ contains n private function symbols $\mathsf{ff}_1,\ldots,\mathsf{ff}_n$ of arity 2.

Consider the following substitution σ_0 with $\mathrm{dom}(\sigma_0) = \cup_{1\leq r\leq k}\widetilde{x}_r$ and $x_i\sigma_0 = \mathsf{ff}_i(z_r, z_d)$ where r is such that $x_i \in \widetilde{x}_r$ and d is given $(d,\cdot) = \mathcal{D}(r,p)$ with p being the control-point at which x_i first occurs in the instructions of r.

The set of clauses associated with the protocol Π is then

$$\mathcal{C}_\Pi \stackrel{\text{def}}{=} \bigcup_{1\leq r\leq k} \{I(u_r^i\sigma_0) \to I(v_r^i\sigma_0) \mid 1 \leq i \leq p_r\}.$$

The initial knowledge of the intruder IK is given by a set of ground terms, and thus modeled by the set of clauses $\mathcal{C}_{IK}$. The secrecy of a message s is thus represented by the unsatisfiability of the set of clauses $\mathcal{C}_\mathcal{I} \cup \mathcal{C}_{IK} \cup \mathcal{C}_\Pi \cup \{\neg I(s)\}$.

Next, for each i, we rename the function symbol ff_i by n_i or k_i depending on the role it plays in modeling: abstracting a nonce or respectively a key.

Exemple 3.5 *The following set of clauses $\mathcal{C}_{\mathsf{NSpk}}$ model the Needham-Schroeder protocol (see Example 1.28, page 48):*

$$\begin{aligned} &\to I(\{\!\![\mathsf{n}_1(z_a,z_b), z_a]\!\!\}_{\mathsf{ek}(z_b)}) \\ I(\{\!\![\mathsf{n}_1(z_a,z_b), x_{n_b}]\!\!\}_{\mathsf{ek}(z_a)}) &\to I(\{\!\![x_{n_b}]\!\!\}_{\mathsf{ek}(z_b)}) \\ I(\{\!\![y_{n_a}, z_a]\!\!\}_{\mathsf{ek}(z_b)}) &\to I(\{\!\![y_{n_a}, \mathsf{n}_2(z_b,z_a))]\!\!\}_{\mathsf{ek}(a)}) \end{aligned}$$

The first two clauses corresponding to A's role, while the third to B's role. For simplicity, we have omitted the literal $-I(\mathsf{init})$ from the first clauses, and the clause

$$I(\{\!\![\mathsf{n}_2(z_b,z_a)]\!\!\}_{\mathsf{ek}(z_b)}) \to I(\mathsf{stop})$$

(corresponding to the second rule of B's role). This is without loss of generality (w.r.t. to the satisfiability problem) since init *and* stop *are public constants and hence the corresponding clauses (i.e. $I(\mathsf{init})$ and $I(\mathsf{stop})$) would be in $\mathcal{C}_{IK}$.*

Since the clauses can be applied in any order, any number of times, we also abstract away the order of the rules of the protocol. The order could be enforced by associating the clause $I(u_1),\ldots,I(u_i) \to I(v_i)$ (instead of $I(u_i) \to I(v_i)$) to each step i of a role. However, this enhancement does not assure that a rule is not played several times.

Note that all these abstractions are correct w.r.t. the secrecy property, *i.e.* if a protocol is deemed secure using these abstractions then it is secure without abstractions. The converse does not hold, i.e. these abstractions are not complete, as the following example shows.

Exemple 3.6 *Consider the following protocol*

$$\begin{aligned} A \Rightarrow B:&\quad \{\!\!\{N_a, K_1\}\!\!\}_{K_{ab}}, \{\!\!\{N_a, K_2\}\!\!\}_{K_{ab}} \\ B \Rightarrow A:&\quad \{\!\!\{N_a, K_1\}\!\!\}_{K_{ab}} \\ A \Rightarrow B:&\quad K_1, \{\!\!\{s\}\!\!\}_{\langle K_1, K_2\rangle} \end{aligned}$$

in which N_a is a fresh nonce, and K_1, K_2 are two fresh keys and K_{ab} is a long-term shared key between A and B. The agent B just sends back to A the first ciphertext. Next, A just verifies that the first component is the right one (i.e. the nonce N_a that she sent in the first step), but she does not verify the second component (the key K_1). That is, using the model of Chapter 1, the second rule of A's role is $\mathsf{recv}(\{\!\{n_a, y\}\!\}_{\mathsf{k}(z_a,z_b)}), \mathsf{send}(\langle y, \{\!\{s\}\!\}_{\langle k_1,k_2\rangle}\rangle)$, where n_a, k_1, k_2 are the fresh items of A's role. The intruder can obtain, in each sessions, either K_1 or K_2 but not both. And since the secret s is encrypted in each session with a new pair of keys the intruder is not able to obtain the secret. However, in our Horn clause model, the secret can be obtained by the intruder.

The set of clauses modeling the above protocol is

$$\begin{array}{rl} & \to I(\langle \{\!\{\mathsf{n}(z_a,z_b), \mathsf{k}_1(z_a,z_b)\}\!\}_{\mathsf{k}(z_a,z_b)}, \{\!\{\mathsf{n}(z_a,z_b), \mathsf{k}_2(z_a,z_b)\}\!\}_{\mathsf{k}(z_a,z_b)}\rangle) \\ I(\langle y_1, y_2\rangle) & \to I(y_1) \\ I(\{\!\{\mathsf{n}(z_a,z_b), x\}\!\}_{\mathsf{k}(z_a,z_b)}) & \to I(\langle x, \{\!\{s\}\!\}_{\langle \mathsf{k}_1(z_a,z_b), \mathsf{k}_2(z_a,z_b)\rangle}\rangle) \end{array}$$

Here the intruder obtains both keys by using twice the third rule (with the pair of ciphertexts of the first rule as it is, and with the ciphertexts interchanged). This is only possible due to our modeling of fresh items by the same term (in all sessions between the same agents); A wouldn't normally accept a ciphertext from an old session since she verifies the freshness of her nonce.

Other security properties can also be encoded as Horn clauses. B. Blanchet has encoded authentication [Bla02], strong secrecy [Bla04], and some other equivalence properties [BAF05] by expressing them in a pi-like process calculus and translating processes to Horn clauses [AB02].

3.2 A fragment of Horn clauses

We have seen that for some protocol, some intruder capabilities, and some secret message, if the corresponding set of clauses is satisfiable then the secrecy of the message is preserved. However, deciding the satisfiability of a set of clauses is in general undecidable. Several decidable classes do exist [FLHT01]. One of such fragments, which is quite well-suited for modeling security protocols, was identified by V. Cortier and H. Comon in [CLC03a]. Other such fragment(s) tailored for security protocols can by found in [SV05, SV06]. We extend the fragment of [CLC03a] in order to capture some special primitives not considered before in this context. In Sections 3.2.1 and 3.2.2 we present existing fragments, while in Section 3.2.3 we introduce the extension.

A class of clauses is a set of sets of clauses. We denote classes of clauses using the symbol $\mathfrak{C}$. By language abuse we say that a clause C is the class $\mathfrak{C}$, if $\{C\}$ is in the class $\mathfrak{C}$. Also, for a set $\mathcal{C}$ of clauses and a class of clauses $\mathfrak{C}$, we denote $\mathcal{C} \cup \mathfrak{C} \stackrel{\text{def}}{=} \{\mathcal{C} \cup \mathcal{C}' \mid \mathcal{C}' \in \mathfrak{C}\}$. When we say that a set of clauses $\mathcal{C}_0$ belongs to the class of clauses in $\mathfrak{C}$ or $\mathfrak{C}'$, we mean that $\mathcal{C}_0 \in \{\mathcal{C} \cup \mathcal{C}' \mid \mathcal{C} \in \mathfrak{C}, \mathcal{C}' \in \mathfrak{C}'\}$.

3.2.1 Intruder clauses

Note that each of the clauses in Example 3.3, except the one for asymmetric decryption, contains at most one function symbol. That is why we consider the following class of clauses.

Definition 3.7 (class $\mathfrak{C}_I$) *The class $\mathfrak{C}_I$ is the class of Horn clauses of the form:*

$$\pm I(f(x_1, \ldots, x_n)) \vee \bigvee_{j=1}^{m} \pm I(x_{i_j}).$$

3.2.2 Protocol clauses

For simplicity we consider next only two-party protocols.

H. Comon and V. Cortier [CLC03b] have shown (using a different Horn clause encoding) that, for secrecy properties, it is sufficient to verify the correctness of a protocol for only three parties: two honest and one dishonest participants. Hence we consider three agents having their identities represented by the constants a, b and i, where a and b stand for the honest participants, while i stands for the dishonest participant. The initial knowledge of the intruder thus contains the (finite) private data of i, but not that of a and b.

We suppose that all roles have the following two parameters z_a, z_b, and we define the set of substitutions $\{\sigma_i \mid 1 \leq i \leq 6\}$ with $\mathrm{dom}(\sigma_i) = \{z_a, z_b\}$ and $\mathrm{ran}(\sigma_i) \subseteq \{a, b, i\}$ such that $\sigma(z_a) \neq \sigma(z_b)$. Then the set of clauses modeling a protocol is

$$\mathcal{C}'_\Pi \stackrel{\text{def}}{=} \bigcup_{1 \leq i \leq 6} \mathcal{C}_\Pi \sigma_i$$

Note that the variables of $\mathcal{C}'_\Pi$ are exactly those of the protocol (thus excluding parameters and fresh items).

We observe that each of the clauses in the set $\mathcal{C}'_{\mathsf{NSpk}}$ (see Example 3.5) has at most one variable. As noticed in [CLC03a], this is the case of protocols *with single blind copying*, *i.e.* protocols for which, at each step of the protocol, at most one part of the received message is blindly "copied" to the sent message. For example, in the first rule of B's role of the Needham-Schroeder protocol (see page 15), the only blindly copied part is N_a since the other part (i.e. A) is an identity known by the B. Therefore, the second class of clauses we consider is the class of Horn clauses that contain at most one variable.

Definition 3.8 (class $\mathfrak{C}_P$) *The class $\mathfrak{C}_P$ is the class of Horn clauses that contain at most one variable.*

Note that by considering a finite number of agents we can also consider public encryption by adding for each identity the clauses for encryption and decryption. More exactly we can replace the clause $I(\{\![x]\!\}_{\mathsf{ek}(y)}), I(\mathsf{dk}(y)) \rightarrow I(x)$ (which is not in $\mathfrak{C}_I$) with the three clauses $I(\{\![x]\!\}_{\mathsf{ek}_\alpha}), I(\mathsf{dk}_\alpha) \rightarrow I(x)$ where $\alpha \in \{a, b, i\}$ and $\mathsf{ek}_\alpha, \mathsf{dk}_\alpha$ are constants. These three clauses are in $\mathfrak{C}_P$.

Note also that all ground clauses are in $\mathfrak{C}_P$. Hence clauses modeling the initial intruder knowledge and the secrecy property (i.e. $-I(s)$) fall in the class. Thus, for standard Dolev-Yao rules and for protocols with single blind copying, the secrecy problem can be modeled (as we have seen in the previous section) by a set of clauses in the class $\mathfrak{C}_I \cup \mathfrak{C}_P$. The satisfiability of this set of clauses will prove the secrecy property.

H. Comon and V. Cortier [CLC03a] have shown that satisfiability of a set of clauses of $\mathfrak{C}_I \cup \mathfrak{C}_P$ is decidable in 3-EXPTIME, and H. Seidl and K. Verma [SV05] have shown that satisfiability is in fact DEXPTIME-complete.

3.2.3 Extending the intruder power

The aim of the chapter is to extend the decidability result of [CLC03a] to a larger class of clauses, in order to model an extended power of the intruder. Indeed, the set of clauses, described in Example 3.3, represents the capabilities of an intruder, assuming perfect cryptography. In particular, the intruder cannot learn anything from an encrypted message $\{\!\{m\}\!\}_k$, except if he has the inverse key. However, depending on the implementation of the cryptographic primitives, the intruder may be able to deduce more messages. We consider here CBC encryption and blind signatures.

3.2.3.1 Prefix property

Depending on the encryption scheme, an intruder may be able to get from an encrypted message the encryption of any of its prefixes: from a message $\{\!\{x, y\}\!\}_z$, he can deduce the message $\{\!\{x\}\!\}_z$. This is encoded by the clause:

$$C_{pre} \stackrel{\text{def}}{=} -I(\{\!\{\langle x, y\rangle\}\!\}_z) \vee I(\{\!\{x\}\!\}_z)$$

This is for example the case for Cipher Block Chaining (CBC) encryption. We recall that in such a system, the encryption of the message block sequence $P_1P_2 \cdots P_n$ (where some bits may be added to P_n such that every block has the same length) with the key K is $C_1C_2 \cdots C_n$ where $C_0 = IV$ (initialisation vector) and $C_i = \{\!\{C_{i-1} \oplus P_i\}\!\}_K$. The CBC encryption system has the following property:

$$\text{if } C_1C_2 \cdots C_iC_{i+1} \cdots C_n = \{\!\{P_1P_2 \cdots P_iP_{i+1} \cdots P_n\}\!\}_K \text{ then } C_1C_2 \cdots C_i = \{\!\{P_1P_2 \cdots P_i\}\!\}_K$$

That is to say an intruder can get $\{\!\{x\}\!\}_z$ from $\{\!\{x, y\}\!\}_z$ if the length of x is a multiple of the block length used by the cryptographic algorithm. This property can be used to mount attacks on several well-known protocols. For example, we explain in Section 3.4.1 the attack discovered by O. Pereira and J.-J. Quisquater [PQ00] on the Needham-Schroeder symmetric key authentication protocol [NS78].

The prefix property also holds for homomorphic encryption, *i.e.* encryption schemes that verify that $\{\!\{\langle x, y\rangle\}\!\}_k = \langle\{\!\{x\}\!\}_k, \{\!\{y\}\!\}_k\rangle$. This is the case of the ECB (Electronic Code Book) encryption scheme for example, where the encryption of message block sequence $P_1P_2 \cdots P_n$ with the key K is simply the sequence $\{\!\{P_1\}\!\}_K\{\!\{P_2\}\!\}_K \cdots \{\!\{P_n\}\!\}_K$. For such encryption schemes, the clause C_{pre} models only partially the intruder power. Indeed, the intruder is able to recombine messages, an action which is not modeled by the clause.

A drawback of our modeling is that we cannot obtain $\{\!\{\langle x_1, x_2\rangle\}\!\}_y$ from $\{\!\{\langle x_1, \langle x_2, x_3\rangle\rangle\}\!\}_y$ using only the clause C_{pre}. This is mainly due to the use of pairing instead of concatenation. On the other hand, note that considering concatenation leads to a non-deterministic model (see Example 1.22, page 46).

3.2.3.2 Blind signatures

Blind signatures are used in voting protocols like the FOO 92 voting protocol [FOO92, KR05]. The idea of the protocol is that the voter first commits its vote v using a blinding function blind and a random blinding factor r: he sends the message $\mathsf{blind}(v, r)$ together with a signature of the message. The administrator A verifies that the voter has the right to vote and has not voted yet. If it is the case, he signs the message, *i.e.* sends the message $[\![\mathsf{blind}(v, r)]\!]_{\mathsf{sk}_a}$. Note that the administrator does not have access to the vote since it is blinded. Now, the voter can unblind the message, getting $[\![v]\!]_{\mathsf{sk}_a}$, using that $\mathsf{unblind}([\![\mathsf{blind}(v, r)]\!]_{\mathsf{sk}_a}, r) = [\![v]\!]_{\mathsf{sk}_a}$. Then the voter can send its vote to the collector.

The "standard" composition and decomposition properties of blinding are modeled by the following clauses:

$$I(x), I(y) \to I(\mathsf{blind}(x, y))$$
$$I(\mathsf{blind}(x, y)), I(y) \to I(x)$$

Note that these clauses fall into the class $\mathfrak{C}_I$.

The "commutativity" property between blinding and signing can be modeled by the clause:

$$C_{sig} \stackrel{\mathrm{def}}{=} -I([\![\mathsf{blind}(x,y)]\!]_z) \vee -I(y) \vee I([\![x]\!]_z).$$

3.2.3.3 Definition of the class $\mathfrak{C}_S$

First let us note that the clauses C_{pre} and C_{sig} are neither in the class $\mathfrak{C}_I$ nor in the class $\mathfrak{C}_P$. Therefore they cannot be treated by the techniques of [CLC03a, SV05].

In order to extend the intruder power to clauses such as C_{pre} or C_{sig}, we consider the class of *special clauses*, denoted by $\mathfrak{C}_S$.

We assume that the set of function symbols $\mathcal{F}$ contains a special symbol f_0 and that this symbol is the smallest symbol of $\mathcal{F}$ for the ordering $<_{\mathcal{F}}$. This special symbol stands for encryption in the case of the prefix property, and stands for signing in the case of blind signatures.

Definition 3.9 (class $\mathfrak{C}_S$) *The class $\mathfrak{C}_S$ is the class of Horn clauses of the form:*

$$-I(f_0(u[g(y_1,\dots,y_k)],v)) \;\vee\; \bigvee_{i=1}^{p} -I(w_i[g(y_1,\dots,y_k)]) \;\vee\; \bigvee_{l=1}^{q} -I(y_{i_l}) \;\vee\; I(f_0(y_j,z)), \qquad (3.2)$$

where $k > 0$, $\{j, i_1, \dots, i_q\} \subseteq \{1,\dots,k\}$, $p,q \geq 0$, u, w_i are ground contexts, v is a term with $\mathrm{var}(v) = \{z\}$*, $g \neq f_0$, and $I(f_0(u[g(y_1,\dots,y_k)],v))$ is greater (w.r.t. $<$) than any other literal of the clause.*

For example, the clause C_{pre} is obtained when $u = v = z$, $j = 1$, $p = q = 0$, $f_0 = \{\!\!\{\_\}\!\!\}_\_$ and, $g = \langle \_ , \_ \rangle$. The clause C_{sig} is obtained when $u = v = z$, $j = 1$, $p = 0$, $q = 1$, $f_0 = [\![\_]\!]_\_$ and, $g = \mathsf{blind}$. We could also consider for example the clause $-I(\{\!\!\{\langle x,y\rangle\}\!\!\}_z) \vee I(\{\!\!\{y\}\!\!\}_z)$.

This class could also be used to express more complex protocol clauses.

3.3 A decidability result

We show that satisfiability of clauses of $\mathfrak{C}_{IPS} \stackrel{\mathrm{def}}{=} \{C_1 \cup C_2 \cup C_3 \mid C_1 \in \mathfrak{C}_I, C_2 \in \mathfrak{C}_P, C_3 \in \mathfrak{C}_S\}$ is still decidable, under a slight semantical assumption. To get this result we consider a variant of ordered resolution where resolution between clauses of a *saturated* set are forbidden. In Section 3.3.1, we recall the definition of ordered resolution. In Section 3.3.2, we introduce our variant of ordered resolution. We prove our decidability result in Section 3.3.3 and show in Section 3.3.4 that both CBC encryption and blind signatures satisfy the hypotheses of our theorem.

3.3.1 Ordered resolution

We consider a liftable partial ordering $\prec$, total on closed atoms.

Let A and B be two unifiable atoms, $\sigma = \mathrm{mgu}(A,B)$, and C_1 and C_2 be two clauses such that $C_1 = C_1' \vee A$ and $C_2 = C_2' \vee -B$ for some clauses C_1' and C_2'. The *binary resolution rule* is defined by:

$$\frac{C_1' \vee A \quad -B \vee C_2'}{C_1'\sigma \vee C_2'\sigma}$$

The clause $C_1'\sigma \vee C_2'\sigma$ is called *resolvent* of the clauses C_1 and C_2. The atom $A\sigma$ is called the *resolved atom*. We have implicitly supposed here that the clauses C_1 and C_2 do not share variables, which can be obtained by renaming the variables of one of the clauses. Note that, by the definition of clauses, the same literal cannot appear twice in a clause. Indeed, we suppose that the resolution rule contains an implicit factorisation which immediately replaces $L \vee L \vee C$ by $L \vee C$.

The *ordered resolution* (w.r.t. $\prec$) requires that there is no atom in the resolvent greater than the resolved atom. Note that in this case, since $\prec$ is liftable, A and B in the above resolution rule are maximal in $C_1 \vee C_2$.

If $C_1, C_2, \ldots, C_n$ are clauses such that their sets of variables are pairwise disjoint then we note the clause $C_1 \vee C_2 \vee \cdots \vee C_n$ by $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_n$, in order to emphasise this property. Considering a set $\mathfrak{C}$ whose elements are sets of clauses, the *splitting rule* is defined as follows:

$$\mathfrak{C} \ \rightarrow_{spl} \ \left(\mathfrak{C}\backslash\{\mathcal{C}\}\right) \ \cup \ \left\{(\mathcal{C}\backslash\{C_1 \sqcup C_2\}) \cup \{C_1\}\right\} \ \cup \ \left\{(\mathcal{C}\backslash\{C_1 \sqcup C_2\}) \cup \{C_2\}\right\},$$

where $\mathcal{C} \in \mathfrak{C}$ and $C_1 \sqcup C_2 \in \mathcal{C}$ and C_1, C_2 not empty. We write $\mathfrak{C} \Rightarrow_{spl} \mathfrak{C}'$ to say that $\mathfrak{C} \rightarrow^*_{spl} \mathfrak{C}'$ and no application of the splitting rule on $\mathfrak{C}'$ is possible anymore.

It is well known that ordered resolution with splitting is complete for Horn clauses [BG01]. However, while ordered resolution was sufficient to prove decidability of satisfiability for clauses of the classes $\mathfrak{C}_I \cup \mathfrak{C}_P$, this is not the case anymore. Consider for example the ordering $<$ defined in Section 3.1.1.1 (which extends the ordering considered in [CLC03a]). Ordered resolution between the clause C_{pre} and the clause $I(x), I(y) \rightarrow I(\{\!\{x\}\!\}_y)$ yields $I(\langle x, y\rangle), I(z) \rightarrow I(\{\!\{x\}\!\}_z)$. Resolving again this clause with C_{pre} yields $I(\langle\langle x, x'\rangle, y\rangle), I(z) \rightarrow I(\{\!\{x\}\!\}_z)$ and so on. Thus ordered resolution does not terminate. However, we note that deriving the clause $I(\langle x, y\rangle), I(z) \rightarrow I(\{\!\{x\}\!\}_z)$ is useless (w.r.t. the completeness of the resolution) thanks to the clause $I(\langle x, y\rangle) \rightarrow I(x)$. This will be formally proved in Section 3.3.4. In terms of resolution theory [BG01], the set $\mathcal{I} \cup \{C_{pre}\}$, where $\mathcal{I}$[12] is the set of clauses described in Example 3.3, is already saturated. We formalise this notion in the next section.

3.3.2 Our resolution method

A *partial ordered interpretation* $\mathfrak{I}$ is a set of ground literals such that if $A \in \mathfrak{I}$ then $-A \notin \mathfrak{I}$ and conversely, and if $\pm A \in \mathfrak{I}$ and $B \prec A$ then $\pm' B \in \mathfrak{I}$ for some sign $\pm'$. A ground clause C is *false* in $\mathfrak{I}$ if, for every literal $\pm A$ in C, the opposite literal $\mp A$ belongs to $\mathfrak{I}$. A clause C is *unsatisfiable* in the partial interpretation $\mathfrak{I}$ if there exists a ground substitution θ such that all atoms of $C\theta$ are among those of $\mathfrak{I}$ and $C\theta$ is false in the interpretation. A set of clauses is *unsatisfiable* in the partial interpretation $\mathfrak{I}$ if there is a clause in the set that is unsatisfiable in $\mathfrak{I}$.

Definition 3.10 (saturated set of clauses) *A set $\mathcal{S}$ of clauses is* saturated *w.r.t. the ordering $\prec$ if for every resolvent C obtained by ordered resolution from $\mathcal{S}$ and for every partial interpretation $\mathfrak{I}$, if C is unsatisfiable in $\mathfrak{I}$ then $\mathcal{S}$ is unsatisfiable in $\mathfrak{I}$.*

Let $\mathcal{S}$ be a saturated set of clauses. For a set of clauses $\mathcal{T}$ we denote by $\mathrm{Res}_{\prec,\mathcal{S}}(\mathcal{T})$ the set of clauses derived by ordered resolution method with the restriction that we do not apply resolution if the two premises are clauses of $\mathcal{S}$. We may drop the subscripts when they are clear from the context.

We define $\mathrm{Res}^*_{\prec,\mathcal{S}}(\mathcal{T}) \stackrel{\text{def}}{=} \mathcal{T} \cup \mathrm{Res}_{\prec,\mathcal{S}}(\mathcal{T})$. For a class $\mathfrak{C}$ of sets of clauses we denote by $\mathrm{Res}^*_{\prec,\mathcal{S}}(\mathfrak{C}) \stackrel{\text{def}}{=} \{\mathrm{Res}^*_{\prec,\mathcal{S}}(\mathcal{T}) \mid \mathcal{T} \in \mathfrak{C}\}$. Also we write $\mathfrak{C} \Rightarrow_{\prec,\mathcal{S},spl} \mathfrak{C}'$ to say that $\mathrm{Res}^*_{\prec,\mathcal{S}}(\mathfrak{C}) \Rightarrow_{spl} \mathfrak{C}'$. Remark that $\mathfrak{C}'$ is unique.

[12] We have overloaded the symbol $\mathcal{I}$ (previously it was used to denote deduction systems). For the rest of this chapter we use $\mathcal{I}$ only to denote some set of clauses.

For a liftable ordering $\prec$, and a set of clauses $\mathcal{S}$, we denote by $\mathcal{R}_{\prec,\mathcal{S}}$ the ordered resolution method with splitting together with the mentioned restriction. The following result states the refutational completeness of this method:

Proposition 3.11 *For any liftable ordering $\prec$, for any sets $\mathcal{S}$ and $\mathcal{T}$ of clauses, such that $\mathcal{S}$ is saturated w.r.t. $\prec$ and $\mathcal{S} \subseteq \mathcal{T}$, $\mathcal{T}$ is unsatisfiable if and only if $\{\mathcal{T}\} \Rightarrow^*_{\prec,\mathcal{S},spl} \mathfrak{C}$, for some $\mathfrak{C}$ such that every set of clauses in $\mathfrak{C}$ contains the empty clause.*

The proof is a direct consequence of the refutational completeness of the standard strategy since, from the hypothesis that $\mathcal{S}$ is saturated, all inferences performed between clauses from $\mathcal{S}$ are useless.

We extend the presented resolution method with a *tautology elimination rule* and a *subsumption rule*. These rules do not compromise the completeness result of the method.

3.3.3 A decidable class

We consider the ordering $<$ defined in Section 3.1.1.1. By Proposition 3.2, $<$ is a liftable ordering. We apply the resolution method $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ (defined in Section 3.3.2) to sets of clauses $\mathcal{I} \cup \mathcal{S} \cup \mathcal{P}$, with $\mathcal{I} \in \mathfrak{C}_I$, $\mathcal{S} \in \mathfrak{C}_S$, $\mathcal{P} \in \mathfrak{C}_P$. Thanks to Proposition 3.11 this method is refutationally complete. Hence to get decidability we only need to show its termination.

However, our resolution method is still not sufficient to ensure termination for clauses of $\mathfrak{C}_{IPS}$.

Exemple 3.12 *Let $C = -I(u[x]) \vee I(\{\!\!\{x\}\!\!\}_{v[x]})$ such that $\{\!\!\{x\}\!\!\}_{v[x]} \not< u[x]$. Resolving C with C_{pre} gives $C' = -I(u[\langle x', y'\rangle]) \vee I(\{\!\!\{x\}\!\!\}_{v[\langle x',y'\rangle]})$ which can be again resolved with C_{pre}, and so on.*

Thus, we consider an additional slight syntactic restriction. From a protocol point of view, this restriction does not reduce the expressivity of the fragment of clauses under consideration.

Definition 3.13 (well-behaved term, well-behaved clause)
We say that a term is well-behaved *if for any of its subterms of the form $f_0(u, v)$ the following two implications hold:*

- *if* $\mathrm{var}(u) \neq \emptyset$ *then v is a constant;*
- *if* $\mathrm{var}(v) \neq \emptyset$ *then u is a ground term.*

We say that a clause of $\mathfrak{C}_S$ is well-behaved *if the terms v, $u[g(y_1, \ldots, y_k)]$ and $w_i[g(y_1, \ldots, y_k)]$, for all i, (see Definition* 3.9*) are well-behaved.*

We say that a clause C not in $\mathfrak{C}_S$ is well-behaved *if for every literal $\pm I(w)$ of C, w is well-behaved.*

Usually, the terms used in modeling cryptographic protocols are well-behaved. For example, if $\mathcal{S} = \{C_{pre}\}$ (or $\mathcal{S} = \{C_{sig}\}$) (see previous section) then $\mathcal{S} \cup \mathcal{C}'_{\mathsf{NSpk}}$ is well-behaved.

We are now ready to state our main result.

Theorem 3.14 *Let $\mathcal{I}, \mathcal{P}, \mathcal{S}$ be finite sets of clauses included respectively in the classes $\mathfrak{C}_I$, $\mathfrak{C}_P$ and $\mathfrak{C}_S$. If $\mathcal{I} \cup \mathcal{S}$ is saturated and $\mathcal{P} \cup \mathcal{S}$ is well-behaved then the satisfiability of $\mathcal{I} \cup \mathcal{P} \cup \mathcal{S}$ is decidable.*

The rest of the subsection is devoted to the outline of the proof of the theorem. For the sake of clarity, some proofs of intermediate results are postponed to Section 3.3.5.

Our resolution method applied to clauses of the class $\mathfrak{C}_{IPS}$ may create clauses outside the class $\mathfrak{C}_{IPS}$. To obtain an invariant, we introduce the following auxiliary class of clauses. We define $\mathfrak{C}_J$ to be the class of clauses of the form:

$$\bigvee_{i=1}^{r} -I(w_i[g(y_1,\ldots,y_k)]) \ \vee\ \bigvee_{l=1}^{s} -I(y_{i_l}) \ \vee\ I(f_0(y_j,a)),$$

where $k>0$, $r\geq 1$, $s\geq 0$, $\{j,i_1,\ldots,i_s\}\subseteq\{1,\ldots,k\}$, $g\neq f_0$, w_i is a ground context for any i, and a is a constant.

We have that the resolution method $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ applied to any set of clauses of $\mathcal{I}\cup\mathcal{S}\cup\mathfrak{C}_P$ or $\mathfrak{C}_J$ yields a clause in $\mathfrak{C}_P$ or $\mathfrak{C}_J$.

Lemma 3.15 *Let $\mathcal{I}$, $\mathcal{P}'$, $\mathcal{S}$ and $\mathcal{J}$ be sets of unspittable clauses of respectively $\mathfrak{C}_I$, $\mathfrak{C}_P$, $\mathfrak{C}_S$ and, $\mathfrak{C}_J$, such that $\mathcal{I}\cup\mathcal{S}$ is saturated and $\mathcal{P}'$, $\mathcal{S}$, and $\mathcal{J}$ are well-behaved. The application of $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution on $\mathcal{I}\cup\mathcal{S}\cup\mathcal{P}'\cup\mathcal{J}$ produces clauses in $\mathfrak{C}_P$ or $\mathfrak{C}_J$. Moreover, the set of resolvents is well-behaved.*

The proof is done in Section 3.3.5.1.

We define the *depth* of a non-empty clause C to be $\|C\| \stackrel{\text{def}}{=} \max_{L\in C}\|L\|$.

We prove in Section 3.3.5.2 that the depth of clauses obtained applying the $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution does not increase except if they are ground, in which case the depth may double.

Lemma 3.16 *Let C_1 and C_2 be respectively two unsplittable clauses in $\mathfrak{C}_I$, $\mathfrak{C}_S$, $\mathfrak{C}_P$, or $\mathfrak{C}_J$, such that for $i\in\{1,2\}$, if C_i is in $\mathfrak{C}_P$, $\mathfrak{C}_S$ or $\mathfrak{C}_J$ then C_i is well-behaved. The resolvent C derived by $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution satisfies: $\|C\|\leq\max(\|C_1\|,\|C_2\|)$ if C is not ground or if C_1 or C_2 is ground, and $\|C\|\leq 2\max(\|C_1\|,\|C_2\|)$ otherwise (that is, if C is ground, and C_1 and C_2 are not ground).*

These two lemmas allow us to conclude. We denote by $\mathcal{C}_0$ the set of clauses $\mathcal{I}\cup\mathcal{S}\cup\mathcal{P}$ and by $\mathfrak{C}_0$ the class $\{\mathcal{C}_0\}$. For every $i\geq 0$ we define recursively $\mathfrak{C}_{i+1}$ to be the class defined by $\mathfrak{C}_i \Rightarrow_{<,\mathcal{I}\cup\mathcal{S},spl} \mathfrak{C}_{i+1}$. Due to the application of the splitting rule, for any i, the elements of the class $\mathfrak{C}_{i+1}$ are sets of clauses such that each of the clauses in these sets is either a ground literal, or does not contain any ground literal.

Using Lemma 3.15, we obtain by induction that for every i, for every $\mathcal{C}\in\mathfrak{C}_i$, we can write $\mathcal{C}=\mathcal{I}\cup\mathcal{S}\cup\mathcal{P}'\cup\mathcal{J}$, where $\mathcal{P}'$ and $\mathcal{J}$ are well-behaved sets of clauses of the classes $\mathfrak{C}_P$ and $\mathfrak{C}_J$ respectively.

Let $N\stackrel{\text{def}}{=}\max_{C\in\mathcal{C}_0}\|C\|$. Applying now Lemma 3.16 and induction, we deduce that for every i, for every $\mathcal{C}\in\mathfrak{C}_i$, for every $C\in\mathcal{C}$, we have that $\|C\|\leq N$ if C is not ground and $\|C\|\leq 2N$ if C is ground.

From the definition of classes $\mathfrak{C}_I$, $\mathfrak{C}_S$, $\mathfrak{C}_P$ and $\mathfrak{C}_J$ we observe that clauses in sets $\mathcal{C}\in\mathfrak{C}_i$, for every i, have at most $k+1$ variables, where k is the maximal arity of function symbols in $\mathcal{F}$ (indeed, clauses in $\mathfrak{C}_S$ may have $k+1$ variables: $y_1,\ldots,y_k$ and z).

Since there is a finite number of sets of clauses of bounded depth (up to variable renaming), we deduce that the $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution terminates.

With regard to the complexity of this decision procedure we obtain, using a similar argument as in [Cor03], that the satisfiability of the set $\mathcal{I}\cup\mathcal{P}\cup\mathcal{S}$ is decidable in 3-EXPTIME.

3.3.4 Examples

In this section we show that the intruder clauses corresponding to our two examples (CBC encryption and blind signatures) are saturated. This means that we can analyse any protocol encoded in $\mathfrak{C}_P$ in the presence of an extended intruder that has access either to CBC encryption or blind signatures.

We assume a fixed basic set of capabilities for the intruder, modeled by the set $\mathcal{I}_0$ consisting of the clauses in Example 3.3 (except the three clauses concerning asymmetric encryption, which are neither in $\mathfrak{C}_I$, nor in $\mathfrak{C}_P$), and of the two clauses modeling composition and decomposition of blind messages (see page 93).

We first consider the case of the CBC encryption.

Proposition 3.17 *The set $\mathcal{I}_0 \cup \{C_{pre}\}$ is saturated.*

Proof Given a partial interpretation $\mathfrak{I}$ and an atom A belonging to $\mathfrak{I}$, we say that A is *true* (resp. *false*) in $\mathfrak{I}$ if A appears with sign $+$ (resp. with sign $-$).

We consider an ordered resolution between clauses of $\mathcal{I}_0 \cup \{C_{pre}\}$. If both premises are clauses of $\mathcal{I}_0$ then all the resolvents are tautologies. Therefore they are satisfiable for any partial interpretation. The only interesting cases are when one of the premise is C_{pre}.

Consider first that the other premise is $-I(x) \vee -I(y) \vee +I(\{\!\{x\}\!\}_y)$. The resolvent of these two clauses is $C_1 \stackrel{\text{def}}{=} -I(\langle x, y\rangle) \vee -I(z) \vee +I(\{\!\{x\}\!\}_z)$. We consider an arbitrary partial interpretation $\mathfrak{I}$ such that C_1 is unsatisfiable in $\mathfrak{I}$. By definition, there exists a ground substitution θ such that $C_1\theta$ is false in $\mathfrak{I}$. The clause $C_1\theta$ has the form $-I(\langle u, v\rangle) \vee -I(w) \vee +I(\{\!\{u\}\!\}_w)$, where u, v and w are ground terms. Thus the literals $+I(\langle u, v\rangle)$, $+I(w)$ and $-I(\{\!\{u\}\!\}_w)$ are in $\mathfrak{I}$. Also, since $u < \{\!\{u\}\!\}_w$, one of the literals $+I(u)$ or $-I(u)$ must appear in $\mathfrak{I}$. We consider the two cases.

- Either the atom $I(u)$ is true in $\mathfrak{I}$ then the clause $-I(u) \vee -I(w) \vee +I(\{\!\{u\}\!\}_w)$ is false in $\mathfrak{I}$ and it follows that the clause $-I(x) \vee -I(y) \vee +I(\{\!\{x\}\!\}_y)$ is unsatisfiable in $\mathfrak{I}$;
- Or the atom $I(u)$ is false in $\mathfrak{I}$ then the clause $-I(\langle u, v\rangle) \vee +I(u)$ is false in $\mathfrak{I}$, therefore the clause $-I(\langle x, y\rangle) \vee +I(x)$ is unsatisfiable in $\mathfrak{I}$.

In both cases a clause of $\mathcal{I}_0 \cup \{C_{pre}\}$ is unsatisfiable in $\mathfrak{I}$.

In the other cases the ordering does not allow the resolution step. Indeed, if both premises are C_{pre} then the resolvent is $-I(\{\!\{\langle\langle x, y\rangle, y'\rangle\}\!\}_z) \vee +I(\{\!\{x\}\!\}_z)$ and the resolved atom is $\{\!\{\langle x, y\rangle\}\!\}_z$. But $\{\!\{\langle x, y\rangle\}\!\}_z < \{\!\{\langle\langle x, y\rangle, y'\rangle\}\!\}_z$. If the premises are C_{pre} and the decryption clause then the resolvent is $-I(\{\!\{x, y\}\!\}_z) \vee -I(z) \vee +I(x)$ and the resolved atom is $I(\{\!\{x\}\!\}_z)$. But $\{\!\{x\}\!\}_z < \{\!\{x, y\}\!\}_z$.

We conclude that the set $\mathcal{I}_0 \cup \{C_{pre}\}$ is saturated. ■

The same property is true in the blind signature case.

Proposition 3.18 *The set $\mathcal{I}_0 \cup \{C_{sig}\}$ is saturated.*

Proof The proof is similar to the previous one. The only interesting case is when one of the premises is C_{sig}. It must be the case that the second premise is $-I(x) \vee -I(y) \vee +I([\![x]\!]_y)$. The resolvent of these two clauses is $-I(\mathsf{blind}(x, y)) \vee -I(y) \vee -I(z) \vee +I([\![x]\!]_z)$. An analogous reasoning as in the previous lemma shows that if the resolvent is unsatisfiable in some partial interpretation $\mathfrak{I}$ then $\mathcal{I}_0 \cup \{C_{sig}\}$ is unsatisfiable in $\mathfrak{I}$. The intuition is that the resolvent could already be obtained from the clauses $-I(x) \vee -I(z) \vee +I([\![x]\!]_z)$ and $-I(\mathsf{blind}(x, y)) \vee -I(y) \vee +I(x)$. ■

As a consequence of these two propositions and applying Theorem 3.14, we get that for any well-behaved set $\mathcal{P}$ (encoding both a protocol and a security property), the satisfiability of $\mathcal{I}_0 \cup \{C_{pre}\} \cup \mathcal{P}$ (resp. of $\mathcal{I}_0 \cup \{C_{sig}\} \cup \mathcal{P}$) is decidable. Since for example secrecy can be modeled using a ground clause (for example $-I(\mathsf{n}(a,b))$ to express that the intruder should not learn the nonce between a and b), we obtained a (correct but incomplete) procedure for verifying the secrecy of protocols that use the described prefix property or blind signatures.

In addition, in the case of other extensions of the intruder power leading to other sets $\mathcal{S}$ of clauses in $\mathfrak{C}_S$, the saturation of the set $\mathcal{I} \cup \mathcal{S}$ can be easily verified by hand (like in our examples).

3.3.5 Proofs of intermediate results

3.3.5.1 Invariance under resolution

We show in this subsection that our resolution method on a set of clauses in $\mathfrak{C}_I$, $\mathfrak{C}_S$, $\mathfrak{C}_P$, or $\mathfrak{C}_J$ produces a set of resolvents that belongs to the class of clauses in $\mathfrak{C}_P$ or $\mathfrak{C}_J$. We first prove a helpful lemma.

Lemma 3.19 *Let w_1 and w_2 be two well-behaved unifiable terms, and let $\sigma = \mathrm{mgu}(w_1, w_2)$. Then $w\sigma$ is well-behaved for any well-behaved term w.*

Proof Suppose that there is a subterm $f_0(u,v)$ of $w\sigma$ such that $f_0(u,v)$ is not well-behaved. We have the following possibilities for $f_0(u,v)$: either it is a subterm of w, or $f_0(u,v) = f_0(u',v')\sigma$ where $f_0(u',v')$ is a subterm of w, or it is a subterm of $x\sigma$, where $x \in \mathrm{var}(w)$. In the first two cases we obtain immediately that w is not well-behaved, since $f_0(u,v)$ and respectively $f_0(u',v')$ are not well-behaved. In the third case there is a subterm $f_0(u'',v'')$ of w_1 or w_2 such that $f_0(u'',v'')\sigma = f_0(u,v)$. Therefore w_1 or w_2 is not well-behaved. Hence we obtained a contradiction, and so the supposition is false. ∎

Lemma 3.15 *Let $\mathcal{I}$, $\mathcal{P}'$, $\mathcal{S}$ and $\mathcal{J}$ be sets of unspittable clauses of respectively $\mathfrak{C}_I$, $\mathfrak{C}_P$, $\mathfrak{C}_S$ and, $\mathfrak{C}_J$, such that $\mathcal{I} \cup \mathcal{S}$ is saturated and $\mathcal{P}'$, $\mathcal{S}$, and $\mathcal{J}$ are well-behaved. The application of $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution on $\mathcal{I} \cup \mathcal{S} \cup \mathcal{P}' \cup \mathcal{J}$ produces clauses in $\mathfrak{C}_P$ or $\mathfrak{C}_J$. Moreover, the set of resolvents is well-behaved.*

Proof Let C_1 and C_2 be clauses in $\mathcal{I} \cup \mathcal{S} \cup \mathcal{P}' \cup \mathcal{J}$. Let C be a resolvent of C_1 and C_2 with $C = C_1'\sigma \vee C_2'\sigma$, where $C_1 = C_1' \vee L_1$, $C_2 = C_2' \vee L_2$, and $\sigma = \mathrm{mgu}(L_1, L_2)$. We have to prove that the clause C is in the class $\mathfrak{C}_P$ or $\mathfrak{C}_J$. In order to obtain this, we examine all possible cases according to the membership of C_1 and C_2 to the sets $\mathcal{I}$, $\mathcal{S}$, $\mathcal{P}'$, and $\mathcal{J}$.

For $l \in \{0,1\}$, if C_l belongs to $\mathcal{I}$, $\mathcal{S}$ or $\mathcal{J}$ then C_l is written as in the definition of classes $\mathfrak{C}_I$, $\mathfrak{C}_S$, and $\mathfrak{C}_J$, respectively. If $C_l \in \mathcal{P}'$ and C_l is ground then the resolvent is also ground and hence in $\mathfrak{C}_P$ (indeed, since C_l is unsplittable, it is either a ground literal or it does not contain any ground literal). Therefore, in what follows, we suppose that for $C_l \in \mathcal{P}'$, C_l is not ground. Also, in what follows, we may denote a term u having only one variable, say x, by $u(x)$ in order to emphasise this property. Hence for $C_l \in \mathcal{P}'$, we write $C_l = \pm I(s(x)) \vee \bigvee_{i=1}^{m} \pm I(t_i(x))$, where $m \geq 0$, and we assume that the resolution inference is performed on the literal $\pm I(s(x))$. If $C_l \in \mathcal{J}$ then we assume that the literal of C_l upon which resolution is performed is $\pm I(w_1[g(y_1,\ldots,y_k)])$ (indeed it cannot be $I(f_0(y_j,a))$ since $f_0(y_j,a) < w_1[g(y_1,\ldots,y_k)]$; even if the context w_1 is empty we have $f_0 <_{\mathcal{F}} g$).

The case study follows:

1. $C_1, C_2 \in \mathcal{P}'$. The resolvent C has at most one variable, hence C is a clause of $\mathfrak{C}_P$.

2. $C_1 \in \mathcal{P}'$ and $C_2 \in \mathcal{S}$. Then $L_2 = -I(f_0(u[g(y_1,\dots,y_k)], v(z)))$ since L_2 is maximal in C_2. The literal L_1 is $+I(s(x))$. The following two cases are possible:

   - $s(x) = x$. Then, by maximality of $s(x)$ in C_1, we have $C_1 = +I(x)$. Hence the resolvent is an instance of C_1. Since subsumed clauses are eliminated, this case does not produce a new clause.
   - $s(x) = f_0(s_1, s_2)$. By hypothesis, $s(x)$ is well-behaved.

     Suppose that $\mathrm{var}(s_1) \neq \emptyset$. Then $s_2 = a$, for some constant a. We deduce that $v = z$ and $z\sigma = a$. The following three cases are possible: (1) $y_i\sigma = s'_i$, where each s'_i is a subterm of s_1 and by consequence the resolvent is in $\mathfrak{C}_P$; (2) $x\sigma = u'[g(y_1,\dots,y_k)]$, where u' is a subcontext of u, and, in this case, the resolvent is in $\mathfrak{C}_J$; (3) $x\sigma = u'$ where u' is a ground subterm of u, and in this case σ is ground, thus C is in $\mathfrak{C}_P$.

     Suppose now that $\mathrm{var}(s_2) \neq \emptyset$. Then s_1 is a ground term. Thus $y_i\sigma$ is ground for any i. It follows that the resolvent is a clause of $\mathfrak{C}_P$.

     Observe that, in all the cases, $f_0(y_j, z)\sigma$ is well-behaved. Since u, v, and w_i are well-behaved we obtain, following the same line of proof as in Lemma 3.19, that the resolvent is also well-behaved.

3. $C_1 \in \mathcal{P}'$ and $C_2 \in \mathcal{I}$: We have $L_1 = \pm I(s(x))$ and $L_2 = \mp I(f(x_1,\dots,x_n))$. The following two cases are possible:

   - $s(x) = x$. Then, by maximality of $s(x)$ we have $C_1 = \pm I(x)$. Therefore C is subsumed by C_1.
   - $s(x) = f(s_1,\dots,s_n)$, where, for all i, s_i is a subterm of $s(x)$. Hence, for all i, $x_i\sigma = s_i$. So the resolvent is in $\mathfrak{C}_P$.

4. $C_1 \in \mathcal{P}'$ and $C_2 \in \mathcal{J}$: We have $L_1 = \pm I(s(x))$ and $L_2 = \mp I(w_1[g(y_1,\dots,y_k)])$. Again, the following two cases are possible:

   - for all i, $y_i\sigma = s_i$, where s_i is a subterm of $s(x)$. In this case C is in $\mathfrak{C}_P$.
   - $x\sigma = w'[g(y_1,\dots,y_k)]$, where w' is a subcontext of w_1. If $r > 1$ then C is in $\mathfrak{C}_J$. If $r = 1$ then C is of the form $I(f_0(y_j, a)) \vee \bigvee_l I(y_{j_l})$ which splits into clauses of $\mathfrak{C}_P$ and either $I(f_0(y_j, a)) \vee I(y_{j_l})$ (if there is l with $j_l = k$) or $I(f_0(y_j, a))$ (otherwise), which are both again in $\mathfrak{C}_P$.

5. $C_1, C_2 \in \mathcal{I} \cup \mathcal{S}$: the strategy forbids any resolution in this case.

6. $C_1 \in \mathcal{I}$ and $C_2 \in \mathcal{J}$: We have $L_1 = \pm I(f(x_1,\dots,x_n))$ and $L_2 = \mp I(w_1[g(y_1,\dots,y_k)])$. As before, two cases are possible:

   - w_1 is the empty context and $g = f$. The clauses derived by resolution (and possibly splitting) belong to $\mathfrak{C}_P$ if $r = 1$ and to $\mathfrak{C}_J$ if $r > 1$.
   - for all i, $x_i\sigma = w'_i[g(y_1,\dots,y_k)]$, where w'_i is a subcontext of w_1. Hence in this case C is in $\mathfrak{C}_J$.

7. $C_1, C_2 \in \mathcal{S} \cup \mathcal{J}$: Since none of the positive literals in C_1, C_2 is maximal in its clause, the resolution inferences are blocked.

To finish the proof of the lemma we have to show that the resolvent C is well-behaved. This is a consequence of the invariance of the well-behaviour property under application of substitutions, which was proved in the Lemma 3.19. ■

3.3.5.2 Termination of the resolution method

We first give some lemmas which will be used in the proof of termination. The two following lemmas are similar to Proposition 8.5 in §8.2.1.3 of [Cor03].

Lemma 3.20 *Let u and v be two unifiable terms having at most one variable, and let $\sigma = \mathrm{mgu}(u, v)$. Then $\|u\sigma\| \leq \max(\|u\|, \|v\|)$ if σ is not ground or if u_1 or u_2 is ground, and $\|u\sigma\| \leq 2\max(\|u\|, \|v\|)$ otherwise.*

Proof If one of the terms, say u, is ground then $u\sigma = u$ and the inequality follows directly.

We suppose that each term has exactly one variable. We denote by x the variable of u and by y the variable of v.

Remark that $\|u\sigma\| = \max(\|u\|, \|u\|_x + \|x\sigma\| - 1) \leq \|u\| + \|x\sigma\| - 1$ (as $\|u\|_x \leq \|u\|$), and similarly for v. Thus if $x\sigma = x$ or $y\sigma = y$ then we're done.

Suppose that $x\sigma \neq x$ and $y\sigma \neq y$. In this case σ is ground. Consider an arbitrary occurrence q_x of x in u. Then q_x is a also position in v (otherwise $x\sigma = x$).

- If $v|_{q_x}$ is ground then $x\sigma = v|_{q_x}$, and thus $\|u\sigma\| \leq \|u\| + \|v\| - 1 \leq 2\max(\|u\|, \|v\|)$.
- Otherwise, $y \in \mathrm{var}(v|_{q_x})$. Let q_y be a position y in v. Then q_y is also a position in u (otherwise $y\sigma = y$). We have $x \notin \mathrm{var}(u|_{q_y})$ (otherwise u and v would not be unifiable). Then $y\sigma = u|_{q_y}$ and $u|_{q_y}$ is ground. Hence $\|v\sigma\| \leq \|v\| + \|u\| - 1 \leq 2\max(\|u\|, \|v\|)$.

■

Using exactly the same proof technique as in the previous lemma, we can show the following similar result.

Lemma 3.21 *Let u and v be two unifiable terms such that $u = u'[g(y_1, \ldots, y_k)]$ where u' is a ground context, $k > 0$, and v has at most one variable. Let $\sigma = \mathrm{mgu}(u, v)$. Then $\|u\sigma\| \leq \max(\|u\|, \|v\|)$ if σ is not ground or if u or v is ground, and $\|u\sigma\| \leq 2\max(\|u\|, \|v\|)$ otherwise.*

The next lemma bounds the depth of the resolvant of two clauses from the class we are intersted in.

Lemma 3.22 *Let $C_1 = I(u_1) \vee C_1'$, $C_2 = -I(u_2) \vee C_2'$, be two unsplittable clauses of $\mathfrak{C}_P \cup \mathfrak{C}_I \cup \mathfrak{C}_S \cup \mathfrak{C}_J$, such that for $i \in \{1, 2\}$, if C_i is in $\mathfrak{C}_P$, $\mathfrak{C}_S$, or $\mathfrak{C}_J$ then C_i is well-behaved. Let $C = C_1'\sigma \vee C_2'\sigma$ be the resolvent of C_1 and C_2 with $\sigma = \mathrm{mgu}(u_1, u_2)$. Let $\alpha \in \{1, 2\}$, with $\alpha = 1$ if σ is not ground or if C_1 or C_2 is ground, and $\alpha = 2$ otherwise. Then $\|u_1\sigma\| \leq \alpha \max(\|u_1\|, \|u_2\|)$.*

Proof We prove this lemma by performing a similar case study as in the proof of Lemma 3.15. But first observe that, since C_i is unsplittable, C_i is ground if and only if u_i is ground (for $i \in \{1, 2\}$). We suppose that u_1 and u_2 are not ground, since otherwise we trivially have that $\|u_1\sigma\| \leq \max(\|u_1\|, \|u_2\|)$.

1. $C_1, C_2 \in \mathfrak{C}_P$. We can apply Lemma 3.20 directly to obtain that $\|u_1\sigma\| \leq \alpha \max(\|u_1\|, \|u_2\|)$.

2. $C_1 \in \mathfrak{C}_P$ and $C_2 \in \mathfrak{C}_S$. Then $u_2 = f_0(u[g(y_1,\ldots,y_k)],v)$, where u and v are as in Definition 3.9.

   We denote by x the variable of C_1. If $u_1 = x$ then $u_2\sigma = u_2$ and hence the property is clearly satisfied. Suppose $u_1 = f_0(s_1,s_2)$. By hypothesis, u_1 is well-behaved.

   - If $\mathrm{var}(s_1) \neq \emptyset$ then $s_2 = a$, for some constant a. In this case we have $v = z$ and $z\sigma = a$. We have that σ and $\mathrm{mgu}(u_1, u_2[{}^a\!/\!_z])$ are equal on $\{x, y_1,\ldots,y_k\}$. Thus, by Lemma 3.21 applied on u_1 and $u_2[{}^a\!/\!_z]$, we have $\|u_1\sigma\| \leq \alpha\max(\|u_1\sigma\|, \|(u_2[{}^a\!/\!_z])\sigma\|) = \alpha\max(\|u_1\sigma\|, \|u_2\sigma\|)$.
   - If $\mathrm{var}(s_2) \neq \emptyset$ then s_1 is ground.

     Let σ_0 be the restriction of σ on $\{y_1,\ldots,y_k\}$. We have that $u[g(y_1,\ldots,y_k)]\sigma_0 = s_1$ and σ_0 is ground. Also $\mathrm{mgu}(u_1, u_2\sigma_0)$ is the restriction of σ on $\{x, z\}$. Then, applying Lemma 3.20 on u_1 and $u_2\sigma_0$ we obtain that $\|u_1\sigma\| \leq \alpha\max(\|u_1\sigma\|, \|(u_2\sigma_0)\sigma\|) = \alpha\max(\|u_1\sigma\|, \|u_2\sigma\|)$.

3. $C_1 \in \mathfrak{C}_P$ and $C_2 \in \mathfrak{C}_I$. Then $u_2 = f(x_1,\ldots,x_n)$ and $\mathrm{var}(u_1) \subseteq \{x\}$ for some variable x. We have either $u_1 = x$ and $x\sigma = u_2$, or $u_1 \neq x$ and $u_1, u_2\sigma$ are equal up to variable renaming. In both cases $\|u_1\sigma\| = \max(\|u_1\|, \|u_2\|)$.

4. $C_1 \in \mathfrak{C}_P$ and $C_2 \in \mathfrak{C}_J$. Then $\mathrm{var}(u_1) \subseteq \{x\}$ for some variable x, and $u_2 = w_1[g(y_1,\ldots,y_k)]$. We can apply Lemma 3.21 directly to obtain that $\|u_1\sigma\| \leq \alpha\max(\|u_1\|, \|u_2\|)$.

5. $C_1, C_2 \in \mathcal{I} \cup \mathcal{S}$. The strategy forbids any resolution in this case.

6. $C_1 \in \mathcal{I}$ and $C_2 \in \mathcal{J}$. We have $u_1 = f(x_1,\ldots,x_n)$ and $u_2 = w_1[g(y_1,\ldots,y_k)]$. We then have $u_1\sigma, u_2$ are equal up to variable renaming, and thus $\|u_1\sigma\| = \max(\|u_1\|, \|u_2\|)$.

7. $C_1, C_2 \in \mathcal{S} \cup \mathcal{J}$. Since none of the positive literals in C_1, C_2 is maximal in its clause, the resolution inferences are blocked.

■

We now show that that every clause derived by our resolution strategy has its size bounded by (twice) the maximum of the sizes of its premises.

Lemma 3.16 *Let C_1 and C_2 be respectively two unsplittable clauses in $\mathfrak{C}_I$, $\mathfrak{C}_S$, $\mathfrak{C}_P$, or $\mathfrak{C}_J$, such that for $i \in \{1,2\}$, if C_i is in $\mathfrak{C}_P$, $\mathfrak{C}_S$ or $\mathfrak{C}_J$ then C_i is well-behaved. The resolvent C derived by $\mathcal{R}_{<,\mathcal{I}\cup\mathcal{S}}$ resolution satisfies: $\|C\| \leq \max(\|C_1\|, \|C_2\|)$ if C is not ground or if C_1 or C_2 is ground, and $\|C\| \leq 2\max(\|C_1\|, \|C_2\|)$ otherwise (that is, if C is ground, and C_1 and C_2 are not ground).*

Proof Let $C_1 = I(u_1) \vee C_1'$, $C_2 = -I(u_2) \vee C_2'$, and $C = C_1'\sigma \vee C_2'\sigma$ be the resolvent of C_1 and C_2 with $\sigma = \mathrm{mgu}(u_1, u_2)$. Again, C_i is ground if and only if u_i is ground, for $i \in \{1,2\}$.

It suffices to show that for every term w such that $\pm I(w) \in C_1 \vee C_2$ we have $\|w\sigma\| \leq \max(\|w\|, \|u_1\|, \|u_2\|)$ if σ is not ground or if u_1 or u_2 is ground, and $\|w\sigma\| \leq 2\max(\|w\|, \|u_1\|, \|u_2\|)$ otherwise. We suppose next that w is not ground, since for w ground the two inequalities are trivial. In this case, both u_1 and u_2 are not ground (by the unsplittability of clauses C_1 and C_2).

Lemma 3.22 ensures that $\|u_1\sigma\| \leq \max(\|u_1\|, \|u_2\|)$ if σ is not ground or if u_1 or u_2 is ground, and $\|u_1\sigma\| \leq 2\max(\|u_1\|,$
$\|u_2\|)$ otherwise. Recall that $u_1\sigma$ is maximal in C, and thus u_1 is maximal in C_1.

If σ is ground then $\|w\sigma\| \leq \|u_1\sigma\|$, since otherwise it follows that $w\sigma > u_1\sigma$ (which contradicts the maximality of $u_1\sigma$ in C). Thus $\|w\sigma\| \leq \|u_1\sigma\| \leq 2\max(\|u_1\|, \|u_2\|)$, and we're done.

We consider now that σ is not ground. We assume without loss of generality that $\pm I(w)$ is a literal of C_1. We can have one of the following cases:

1. $C_1 \in \mathfrak{C}_P$. Let x be the variable of C_1. We compare the depth of w with that of u_1.

   - Case $\|w\| > \|u_1\|$. By maximality of u_1 in C_1, $\|w\|_x \leq \|u_1\|_x$. We have that

     $$\|w\sigma\| = \max(\|w\|, \|w\|_x + \|x\sigma\| - 1) \leq \max(\|w\|, \|u_1\|_x + \|x\sigma\| - 1) \leq \max(\|w\|, \|u_1\sigma\|).$$

   - Case $\|w\| \leq \|u_1\|$. If $\|w\|_x \leq \|u_1\|_x$ then $\|w\sigma\| \leq \|u_1\sigma\|$. If $\|w\|_x > \|u_1\|_x$ then for all $y \in \mathrm{var}(x\sigma)$, $\|w\sigma\|_y > \|u_1\sigma\|_y$. Therefore $\|w\sigma\| \leq \|u_1\sigma\|$ because otherwise $w\sigma > u_1\sigma$.

   In both cases $\|w\sigma\| \leq \max(\|w\|, \|u_1\sigma\|)$. Thus, $\|w\sigma\| \leq \max(\|w\|, \|u_1\|, \|u_2\|)$.

2. $C_1 \in \mathfrak{C}_I$. Then $w < u_1$ since $u_1 = f(x_1, \ldots, x_n)$ and $w = x_i$ for some $1 \leq i \leq n$.

3. $C_1 \in \mathfrak{C}_S$. Then $w < u_1$ by the definition of the class $\mathfrak{C}_S$ (see Definition 3.9).

4. $C_1 \in \mathfrak{C}_J$. Then $u_1 = w'[g(y_1, \ldots, y_k)]$ for some closed context w'. If $w = f_0(y_j, a)$ or $w = y_{i_l}$ then $w < u_1$. Suppose that $w = w''[g(y_1, \ldots, y_k)]$. Observe that for $v \in \{w, u_1\}$, the quantity $\|v\|_{y_j}$ is the same for all j. Also note that $\mathrm{var}(w\sigma) = \mathrm{var}(u_1\sigma) = \cup_j \mathrm{var}(y_j\sigma)$. Then by performing the same analysis as in the case $C_1 \in \mathfrak{C}_P$ (by replacing x with y_j for some j), we obtain that $\|w\sigma\| \leq \max(\|w\|, \|u_1\|, \|u_2\|)$.

■

3.4 Application to the Needham-Schroeder symmetric key protocol

3.4.1 Presentation of the protocol

We consider the Needham-Schroeder symmetric key authentication protocol [NS78] as an example of application of our result. The goal of the protocol is the key exchange between two parties, which we call Alice and Bob, and the mutual conviction of the possession of the key by each other. The key is created by a trusted server which shares the secret keys K_{as} and K_{bs} with Alice and Bob respectively. The description of the protocol is as follows:

$$P_{\mathsf{NS}} : \left\{ \begin{array}{ll} A \Rightarrow S : & A, B, N_a \\ S \Rightarrow A : & \{\!\!\{N_a, B, K_{ab}, \{\!\!\{K_{ab}, A\}\!\!\}_{K_{bs}}\}\!\!\}_{K_{as}} \\ A \Rightarrow B : & \{\!\!\{K_{ab}, A\}\!\!\}_{K_{bs}} \\ B \Rightarrow A : & \{\!\!\{N_b\}\!\!\}_{K_{ab}} \\ A \Rightarrow B : & \{\!\!\{N_b - 1\}\!\!\}_{K_{ab}} \end{array} \right.$$

Here we concentrate on the key exchange goal, rather than on the authentication of the two parties. The key exchange goal can be expressed as the secrecy of the nonce N_b. Intuitively, if N_b remains secret, it means that the key K_{ab} used by B has also been kept secret.

If the encryption scheme used to implement this protocol is used in the CBC mode then the following attack [PQ00] is possible. In a first session (1), an intruder can listen to the message

$\{\!\!\{N_a, B, K_{ab}, \{\!\!\{K_{ab}, A\}\!\!\}_{K_{bs}}\}\!\!\}_{K_{as}}$ and then, using the prefix property, he can compute $\{\!\!\{N_a, B\}\!\!\}_{K_{as}}$. In another session of the protocol, he can send it to Alice in the third round. Alice thinks that Bob has started a session (2) with her: Bob plays the role of the initiator and Alice the role of the second participant. And so Alice would use N_a as the shared key, while it is a publicly known message. This attack is summarised in Figure 3.1.

$$\begin{array}{lll}
(1).1 & A \Rightarrow S: & A, B, N_a \\
(1).2 & S \Rightarrow A: & \{\!\!\{N_a, B, K_{ab}, \{\!\!\{K_{ab}, A\}\!\!\}_{K_{bs}}\}\!\!\}_{K_{as}} \\
(2).3 & I(B) \Rightarrow A: & \{\!\!\{N_a, B\}\!\!\}_{K_{as}} \\
(2).4 & A \Rightarrow I(B): & \{\!\!\{N'_a\}\!\!\}_{N_a}
\end{array}$$

Figure 3.1: Attack on the Needham-Schroeder protocol, using the prefix property

The clauses that model the protocol rules are the following ones:

$$\begin{array}{rl}
& \rightarrow I(\langle z_a, z_b, \mathsf{n}_1(z_a, z_b)\rangle) \\
I(\langle z_a, z_b, x\rangle) & \rightarrow I(\{\!\!\{x, z_b, \mathsf{k}(z_a, z_b), \{\!\!\{\mathsf{k}(z_a, z_b), z_a\}\!\!\}_{\mathsf{k}(z_b,z_s)}\}\!\!\}_{\mathsf{k}(z_a,z_s)}) \\
I(\{\!\!\{\mathsf{n}_1(z_a, z_b), z_b, y, z\}\!\!\}_{\mathsf{k}(z_a,z_s)}) & \rightarrow I(z) \\
I(\{\!\!\{y, z_a\}\!\!\}_{\mathsf{k}(z_b,z_s)}) & \rightarrow I(\{\!\!\{\mathsf{n}_2(z_b, z_a)\}\!\!\}_y) \\
I(\{\!\!\{x\}\!\!\}_y) & \rightarrow I(\{\!\!\{\mathsf{pred}(x)\}\!\!\}_y)
\end{array}$$

where pred is public function symbol of sort $\mathsf{Nonce} \rightarrow \mathsf{Nonce}$. Then the protocol is modeled as in Section 3.2.2 by the set of clause $\mathcal{C}'_{\mathsf{NS}}$ obtained from the above clause by instantiating the parameters z_a and z_b by the constants a, b, and i, and z_s by s.

The intruder has also some initial knowledge. He knows the identities of the participating parties, he can create nonces and, he knows the secret key of the compromised agent. This initial knowledge is modeled by the following clauses:

$$\begin{array}{ll}
\rightarrow I(a) & \rightarrow I(\mathsf{k}(i, s)) \\
\rightarrow I(b) & \rightarrow I(\mathsf{n}_1(i, x)) \\
\rightarrow I(i) & \rightarrow I(\mathsf{n}_2(i, x))
\end{array}$$

We denote this set of clauses, corresponding to the initial knowledge of the intruder, by $\mathcal{P}_0$. We remark that these clauses are either ground or with a single variable, and thus belong to $\mathfrak{C}_P$.

In addition, we enrich the set $\mathcal{I}$ (i.e. the clauses of Example 3.3) with the clause $I(x) \rightarrow I(\mathsf{pred}(x))$ that models the ability of the intruder to compute the predecessor of a message (seen as a number). We denote by $\mathcal{I}'$ this enriched set.

3.4.2 Correcting the protocol

We remark that the attack comes from the fact that the intruder, using the second rule of the protocol together with the CBC property, can get the encryption of any message by the key K_{as}: replacing the nonce N_a by any plaintext m of its choice in the first message, he obtains a message of the form $\{\!\!\{m, \ldots\}\!\!\}_{K_{as}}$ from the server and using the CBC property he gets $\{\!\!\{m\}\!\!\}_{K_{as}}$.

To avoid this, we interchange the place of N_a and B in the message sent in the second round. But a similar attack is still possible since the intruder can modify the first message of Alice and send $\langle A, B, B\rangle$ to the server. Then the shared key would be the identity B. Such an attack is possible only if identities can be confused with keys.

To avoid such a type flaw attack, we add a hash of the shared key as the first component of the message sent by the server to Alice and then to Bob. Note that this second transformation is not sufficient by itself (i.e. without interchanging N_a and B) since the intruder has also the ability to produce hashes. The obtained protocol is described below. We refer to this version as the corrected protocol.

$$P_{\mathsf{NSc}} : \begin{cases} A \Rightarrow S : & A, B, N_a \\ S \Rightarrow A : & \{\!\{B, N_a, K_{ab}, \{\!\{h(K_{ab}), K_{ab}, A\}\!\}_{K_{bs}}\}\!\}_{K_{as}} \\ A \Rightarrow B : & \{\!\{h(K_{ab}), K_{ab}, A\}\!\}_{K_{bs}} \\ B \Rightarrow A : & \{\!\{N_b\}\!\}_{K_{ab}} \\ A \Rightarrow B : & \{\!\{N_b - 1\}\!\}_{K_{ab}} \end{cases}$$

The clauses that model the rules of this protocol are the following ones:

$$\begin{array}{rl} & \to I(\langle z_a, z_b, \mathsf{n}_1(z_a, z_b)\rangle) \\ I(\langle z_a, z_b, x\rangle) & \to I(\{\!\{z_b, x, \mathsf{k}(z_a, z_b), \{\!\{\mathsf{h}(\mathsf{k}(z_a, z_b)), \mathsf{k}(z_a, z_b), z_a\}\!\}_{\mathsf{k}(z_b,s)}\}\!\}_{\mathsf{k}(z_a,s)}) \\ I(\{\!\{z_b, \mathsf{n}_1(z_a, z_b), y, z\}\!\}_{\mathsf{k}(z_a,s)}) & \to I(z) \\ I(\{\!\{\mathsf{h}(y), y, z_a\}\!\}_{\mathsf{k}(z_b,s)}) & \to I(\{\!\{\mathsf{n}_2(z_b, z_a)\}\!\}_y) \\ I(\{\!\{x\}\!\}_y) & \to I(\{\!\{\mathsf{pred}(x)\}\!\}_y) \end{array}$$

As for the protocol P_{NS}, we denote by $\mathcal{C}_{\mathsf{NSc}}$ the set of clauses modeling the protocol NSc as in Section 3.2.2.

The aim of the rest of the section is to prove that the corrected protocol preserves the secrecy of N_b.

3.4.3 A transformation preserving secrecy

We observe that the clauses corresponding to the third round and fifth round of the protocol P_{NSc} are not in $\mathfrak{C}_P$ since they have two variables. Therefore we cannot apply directly our result and we are led to an additional modification of the protocol.

We remark that the server sends to Alice in the second round an encrypted message that Alice cannot decrypt. This message could be directly sent to Bob by the server. In addition, the last rule of the protocol does not seem to be able to compromise the secrecy of N_b, thus we remove it. These modifications yield the following protocol:

$$P_{\mathsf{NSv}} : \begin{cases} A \Rightarrow S : & A, B, N_a \\ S \Rightarrow A : & \{\!\{B, N_a, K_{ab}\}\!\}_{K_{as}} \\ S \Rightarrow B : & \{\!\{h(K_{ab}), K_{ab}, A\}\!\}_{K_{bs}} \\ B \Rightarrow A : & \{\!\{N_b\}\!\}_{K_{ab}} \end{cases}$$

The set of clauses that model the protocol are listed below:

$$\begin{array}{rl} & \to I(\langle z_a, z_b, \mathsf{n}_1(z_a, z_b)\rangle) \\ I(\langle z_a, z_b, x\rangle) & \to I(\{\!\{z_b, x, \mathsf{k}(z_a, z_b)\}\!\}_{\mathsf{k}(z_a,s)}) \\ I(\langle z_a, z_b, x\rangle) & \to I(\{\!\{\mathsf{h}(\mathsf{k}(z_a, z_b)), \mathsf{k}(z_a, z_b), z_a\}\!\}_{\mathsf{k}(z_b,s)}) \\ I(\{\!\{\mathsf{h}(y), y, z_a\}\!\}_{\mathsf{k}(z_b,s)}) & \to I(\{\!\{\mathsf{n}_2(z_b, z_a)\}\!\}_y) \end{array}$$

As before, we denote by $\mathcal{C}_{\mathsf{NSv}}$ the set of clauses modeling the protocol NSv as in Section 3.2.2.

Our approach is as follows: we prove that this version is a weaker version than the corrected protocol, *i.e.* that its correctness implies the correctness of the corrected version. Then, since this version fits our class, we apply our resolution method to prove that this version preserves the secrecy of N_b, which allow us to conclude that the corrected version also preserves the secrecy of N_b.

For each protocol P_l, where l is NS, NSc or NSv, we note by $\mathcal{T}_l \stackrel{\text{def}}{=} \mathcal{I}' \cup \mathcal{P}_0 \cup \mathcal{C}'_l \cup \{C_{pre}\}$ the entire set of clauses that model the protocol ($\mathcal{C}_l$ is the set of clauses representing only the rounds of the protocol). The secrecy property of the protocol P_l can be formulated as the satisfiability of the set of clauses $\mathcal{T}_l \cup \{-I(\mathsf{n}_2(b,a))\}$.

We have already seen that $\mathcal{T}_{\mathsf{NS}} \cup \{-I(\mathsf{n}_2(b,a))\}$ is not satisfiable. We prove that the satisfiability of $\mathcal{T}_{\mathsf{NSc}} \cup \{-I(\mathsf{n}_2(b,a))\}$ can be reduced to the satisfiability of $\mathcal{T}_{\mathsf{NSv}} \cup \{-I(\mathsf{n}_2(b,a))\}$.

Proposition 3.23 *If the set of clauses $\mathcal{T}_{\mathsf{NSv}} \cup \{-I(\mathsf{n}_2(b,a))\}$ is satisfiable then the set of clauses $\mathcal{T}_{\mathsf{NSc}} \cup \{-I(\mathsf{n}_2(b,a))\}$ is also satisfiable.*

To prove this proposition, we use another variant of the resolution method, the *positive resolution* [BG01], which requires that one of the premise is a positive clause (i.e. a clause having only positive literals). The method is also refutationally complete. Since we consider Horn clauses, the set $\mathcal{T}_l \cup \{-I(\mathsf{n}_2(b,a))\}$ is unsatisfiable if and only if there is a deduction of the clause $+I(\mathsf{n}_2(b,a))$ by positive resolution on $\mathcal{T}_l$. We denote by $P_l \Vdash I(m)$ the fact that the clause $+I(m)$ can be obtained by positive resolution on $\mathcal{T}_l$.

The following property ensures that the transformation of protocol P_{NSc} in P_{NSv} preserves the secrecy. In other words, if there is an attack in P_{NSc} then there is a corresponding attack in P_{NSv}.

Proposition 3.24 *If $P_{\mathsf{NSc}} \Vdash I(\mathsf{n}_2(b,a))$ then $P_{\mathsf{NSv}} \Vdash I(\mathsf{n}_2(b,a))$.*

Proof To prove the proposition, it is sufficient to find an application $t \mapsto \overline{t}$ on the set of ground terms such that $\overline{\mathsf{n}_2(b,a)} = \mathsf{n}_2(b,a)$ and, for all message m, if $P_{\mathsf{NSc}} \Vdash I(m)$ then $P_{\mathsf{NSv}} \Vdash I(\overline{m})$. We show that the following application satisfies the required properties.

$$
\begin{aligned}
\overline{a} &= a, \text{ for all constant } a\\
\overline{x} &= x, \text{ for all variable } x\\
\overline{\{\!\!\{u\}\!\!\}_v} &= \begin{cases} \langle \{\!\!\{a,\overline{t},\overline{t'}\}\!\!\}_{\mathsf{k}(a',s)}, \overline{t''}\rangle & \text{if } \{\!\!\{u\}\!\!\}_v = \{\!\!\{a,t,t',t''\}\!\!\}_{\mathsf{k}(a',s)},\\ \mathsf{n}_1(i,i) & \text{if } u = \mathsf{pred}(r),\\ \{\!\!\{\overline{u}\}\!\!\}_{\overline{v}} & \text{otherwise.}\end{cases}\\
\overline{\mathsf{pred}(u)} &= \mathsf{n}_1(i,i)\\
\overline{f(u_1,\ldots,u_n)} &= f(\overline{u_1},\ldots,\overline{u_n}), \forall f \in \mathcal{F}, f \neq \{\!\!\{\_\}\!\!\}_\_, f \neq \mathsf{pred}
\end{aligned}
$$

In what follows, a, b are arbitrary constants, r, t are arbitrary terms, while i and s are fixed constants, standing for the intruder and server identities respectively.

We only need to consider deductions of the form:

$$
\frac{C \stackrel{\text{def}}{=} \bigvee_{i=1}^{n} -I(u_i) \vee +I(u) \quad +I(u_1) \ \cdots \ +I(u_n)}{+I(u)}
$$

where C is an instance of a clause of $\mathcal{T}_{\mathsf{NSc}}$. Thus we are reduced to show that, for each ground instance $\bigvee_{i=1}^{n} -I(m_i) \vee +I(m)$ of a clause C of $\mathcal{T}_{\mathsf{NSc}}$, if $P_{\mathsf{NSv}} \Vdash I(\overline{m_i})$, for every i, then $P_{\mathsf{NSv}} \Vdash I(\overline{m})$. We only present here the more interesting cases.

- $C = -I(x) \vee -I(y) \vee +I(\{\!\!\{x\}\!\!\}_y)$. We have to verify that if $P_{\mathsf{NSv}} \Vdash I(\overline{u})$ and $P_{\mathsf{NSv}} \Vdash I(\overline{v})$, where u and v are two ground terms, then $P_{\mathsf{NSv}} \Vdash I(\overline{\{\!\!\{u\}\!\!\}_v})$.

  Suppose that $\{\!\!\{u\}\!\!\}_v$ is of the form $\{\!\!\{a,t,t',t''\}\!\!\}_{\mathsf{k}(a',s)}$. Then we have that $P_{\mathsf{NSv}} \Vdash I(\langle a,\overline{t},\overline{t'},\overline{t''}\rangle)$ and $P_{\mathsf{NSv}} \Vdash I(\mathsf{k}(a',s))$. The projection clauses are in T_{NSv}. Using them with the first relation we obtain $P_{\mathsf{NSv}} \Vdash I(\langle a,\overline{t},\overline{t'}\rangle)$ and $P_{\mathsf{NSv}} \Vdash I(\overline{t''})$. Now using the encryption clause and then the pairing clause we obtain that $P_{\mathsf{NSv}} \Vdash I(\langle \overline{\{\!\{a, \overline{t}, \overline{t'}\}\!\}_{\mathsf{k}(a',s)}}, \overline{t''}\rangle)$, which is what we needed.

Suppose now that $u = \mathsf{pred}(r)$. Then we have $\overline{\{\!\{u\}\!\}_v} = \mathsf{n}_1(i,i)$. But $P_{\mathsf{NSv}} \Vdash \overline{\mathsf{n}_1(i,i)}$, as $+I(\mathsf{n}_1(i,i))$ is a clause from $\mathcal{P}_0$.

If we are in none of these two special cases then it is sufficient to use the encryption clause in order to obtain that $P_{\mathsf{NSv}} \Vdash I(\overline{\{\!\{u\}\!\}_v})$.

- $C = -I(\{\!\{x\}\!\}_y) \vee -I(y) \vee +I(x)$. We have to show that if $P_{\mathsf{NSv}} \Vdash I(\overline{\{\!\{u\}\!\}_v})$ and $P_{\mathsf{NSv}} \Vdash I(\overline{v})$, where u and v are two ground terms, then $P_{\mathsf{NSv}} \Vdash I(\overline{u})$.

  If $\{\!\{u\}\!\}_v = \{\!\{a, t, t', t''\}\!\}_{\mathsf{k}(a',s)}$ then we have $P_{\mathsf{NSv}} \Vdash I(\langle \overline{\{\!\{a, \overline{t}, \overline{t'}\}\!\}_{\mathsf{k}(a',s)}}, \overline{t''}\rangle))$. Therefore we obtain $P_{\mathsf{NSv}} \Vdash I(\overline{\{\!\{a, \overline{t}, \overline{t'}\}\!\}_{\mathsf{k}(a',s)}})$ and $P_{\mathsf{NSv}} \Vdash I(\overline{t''})$. But we also have $P_{\mathsf{NSv}} \Vdash I(\mathsf{k}(a',s))$. And, as the decryption clause is in the model of P_{NSv}, we obtain $P_{\mathsf{NSv}} \Vdash I(\langle a, \overline{t}, \overline{t'}\rangle)$. From which we arrive at the desired relation $P_{\mathsf{NSv}} \Vdash I(\langle a, \overline{t}, \overline{t'}, \overline{t''}\rangle)$.

  If $u = \mathsf{pred}(r)$ then there is nothing to prove because $\overline{\mathsf{pred}(u)} = \mathsf{n}_1(i,i)$ and $P_{\mathsf{NSv}} \Vdash I(\mathsf{n}_1(i,i))$. Otherwise, the proof is direct.

- $C = -I(\langle a, b, x\rangle) \vee +I(\{\!\{b, x, \mathsf{k}(a,b), \{\!\{\mathsf{h}(\mathsf{k}(a,b)), \mathsf{k}(a,b), a\}\!\}_{\mathsf{k}(b,s)}\}\!\}_{\mathsf{k}(a,s)})$. Knowing that $P_{\mathsf{NSv}} \Vdash I(\overline{\langle a, b, u\rangle})$, where u is a ground term, we must obtain that the transformed positive literal of C is deductible from $\mathcal{T}_{\mathsf{NSv}}$.

  The second clause of $\mathcal{C}_{\mathsf{NSv}}$ assures that we have $P_{\mathsf{NSv}} \Vdash I(\{\!\{b, \overline{u}, \mathsf{k}(a,b)\}\!\}_{\mathsf{k}(a,s)})$. Applying the pairing clause and the third clause of $\mathcal{C}_{\mathsf{NSv}}$, we obtain what we needed, *i.e.* $P_{\mathsf{NSv}} \Vdash I(\langle \{\!\{b, \overline{u}, \mathsf{k}(a,b)\}\!\}_{\mathsf{k}(a,s)}, \{\!\{\mathsf{h}(\mathsf{k}(a,b)), \mathsf{k}(a,b), a\}\!\}_{\mathsf{k}(b,s)}\rangle)$.

- $C = -I(\{\!\{b, \mathsf{n}_1(a,b), y, z\}\!\}_{\mathsf{k}(a,s)}) \vee +I(z)$. For any two ground terms u and v, we must prove that if $P_{\mathsf{NSv}} \Vdash I(\overline{\{\!\{b, \mathsf{n}_1(a,b), u, v\}\!\}_{\mathsf{k}(a,s)}})$ then $P_{\mathsf{NSv}} \Vdash I(\overline{v})$. But this is immediate, from the definition of the application and by using the projection on the second component.

- $C = -I(\{\!\{x\}\!\}_y) \vee +I(\{\!\{\mathsf{pred}(x)\}\!\}_y)$. As we have $P_{\mathsf{NSv}} \Vdash I(\mathsf{n}_1(i,i))$ and, for all ground terms u and v, $\overline{\{\!\{\mathsf{pred}(u)\}\!\}_v} = \mathsf{n}_1(i,i)$, this case is trivial.

The conclusion in the remaining cases follows directly from the definition of the application $t \mapsto \overline{t}$. ■

3.4.4 Secrecy of the corrected protocol

Since the clauses of $\mathcal{T}_{\mathsf{NSv}}$ satisfy the hypotheses of our main result, we have verified using our resolution method that the transformed protocol P_{NSv} has no attack. The verification was done automatically using a prototype implementation of the procedure in [CLC03a] that we have extended for our resolution method.

Proposition 3.25 *The set of clauses $\mathcal{T}_{\mathsf{NSv}} \cup \{-I(\mathsf{n}_2(b,a))\}$ is satisfiable.*

We can state now the correctness of the protocol P_{NSc}.

Corollary 3.26 *The set of clauses $\mathcal{T}_{\mathsf{NSc}} \cup \{-I(\mathsf{n}_2(b,a))\}$ is satisfiable.*

Proof Immediate, by Propositions 3.23 and 3.25. ■

3.5 Conclusions

We have obtained new decidability results for the secrecy of cryptographic protocols that employ encryption primitives satisfying properties that could not be treated by previous decision procedures (modulo the approximations introduced by Horn clauses). The results followed from the termination of a resolution strategy on a class of Horn clauses. This resolution strategy might be useful for larger classes of protocols and more encryption properties. Indeed, while termination is no more ensured for larger classes, completeness is still guaranteed.

We have applied our technique to the debugging of a protocol under a more realistic threat model than the one usually considered. We have transformed this protocol so that it falls into the scope of our Horn class. This transformation preserves the attacks and therefore the correctness of the target protocol ensures the correctness of the initial one. The transformation is interesting in itself. We would like to further investigate this type of transformations and to characterise the protocols to which they can be safely applied.

We have used a prototype implementation to automatically test the correctness of protocols presented in this chapter. We would like to further develop it, optimise it, and test it against a library of protocols like [CJ97].

Part II

Transfer results

Chapter 4

From simple secrecy to strong secrecy

Contents

4.1 The model **113**
- 4.1.1 The applied pi calculus 113
- 4.1.2 Modeling protocols within the applied pi calculus 115
- 4.1.3 Secrecy properties 118

4.2 Passive case **120**
- 4.2.1 Simple secrecy implies strong secrecy 120
- 4.2.2 Generalisation of well-formed frames 122

4.3 Active case **126**
- 4.3.1 Our hypotheses 126
- 4.3.2 Main result 130
- 4.3.3 Proofs of intermediate results 132

4.4 Application to some cryptographic protocols **135**
- 4.4.1 Yahalom 135
- 4.4.2 Needham-Schroeder symmetric key protocol 135
- 4.4.3 Wide Mouthed Frog Protocol (modified) 136

4.5 Conclusions **137**

As we have seen in the Introduction, two styles of definitions are usually considered to express that a security protocol preserves the confidentiality of a data s. Simple secrecy says that the secret is never accessible to the adversary. For example, consider the following protocol where the agent A simply sends a secret s to an agent B, encrypted with B's public key.

$$A \to B : \{\mathsf{s}\}_{\mathsf{pub}(B)}$$

An intruder cannot deduce s, thus s is a simple[13] secret. Although this notion of secrecy may be sufficient in many scenarios, in others, stronger security requirements are desirable. For instance consider a setting where s is a vote and B behaves differently depending on its value. If the actions of B are observable, s remains a simple secret but an attacker can learn the values of the vote by watching B's actions. The design of equivalence-based secrecy is targeted at such scenarios and intuitively says that an adversary cannot observe the difference when the value of

[13] By language abuse, we overload here the meaning of "simply" which here refers to the fact that s is a simple secret.

the secret changes. This definition is essential to express properties like confidentiality of a vote, of a password, or the anonymity of participants to a protocol.

Although the second formulation ensures a higher level of security and is closer to cryptographic notions of secrecy, so far decidability results and automatic tools have mainly focused on the first definition.

Related work on strong secrecy Many works have been dedicated to proving correctness properties of protocols such as strong secrecy using contextual equivalences on process calculi, like the spi calculus [AG97]. In particular framed bisimilarity has been introduced by M. Abadi and A. Gordon [AG98] for this purpose. However it was not well suited for automation, as the definition of framed bisimilarity uses several levels of quantification over infinite domains (e.g. set of contexts). In [EHHN99] the authors introduce fenced bisimilarity as an attempt to eliminate one of the quantifiers. Another approach to circumvent the context quantification problems is presented in [BNP99] where labeled transition systems are constrained by the knowledge the environment has of nonces and keys. This approach allows for more direct proofs of equivalence. Similarly, in [BBN04], J. Borgström *et al* propose a sound but incomplete decision procedure based on a symbolic bisimulation. In [DSV00] model-checking techniques for the verification of spi calculus testing equivalence are explored. The technique is limited to finite processes but seems to perform well on some examples. The concept of logical relations for the polymorphic lambda calculus has also been employed to prove behavioural equivalences between programs that rely on encryption in a compositional manner [SP03].

We should mention here some related works based on the concept of non-interference [GM82]. This notion formalises the absence of unauthorised information flow in multilevel computer systems. Non-interference has been widely investigated in the context of language-based security (e.g. [VIS96, ZM01]). It can be expressed with process equivalence techniques and has been applied also to security protocols in [FGM00, BCR03]. An advantage of this approach is that various security properties, including secrecy, can be modeled by selecting proper equivalence relations. However, as far as we know, decidability results for non-interference properties of security protocols have not been reported.

Despite the efforts towards automatically checking of equivalence-based properties for security protocols, the only tool capable of proving strong secrecy for un unbounded number of sessions is the resolution-based algorithm of ProVerif [Bla04] that has been extended for this purpose. ProVerif has also been enhanced [BAF05] for handling equivalences of processes that differ only in the choice of some terms in the context of the applied pi calculus [AF01]. However, in ProVerif termination is not ensured in general.

Finally, very few decidability results are available for strong secrecy. In [Hüt02], H. Hüttel proves decidability for a finite spi calculus (i.e. no replication, thus a bounded number of sessions) for framed bisimilarity. Considering finite processes too, this time in an extension of the applied pi calculus, M. Baudet gives a procedure [Bau05, Bau07] for deciding equivalence-based properties (mainly strong secrecy and resistance to guessing attacks).

Outline of the chapter In this chapter we investigate the situations where simple secrecy entails strong secrecy. We first show that in the passive case (§4.2), reachability-based secrecy actually implies equivalence-based secrecy provided that encryption is probabilistic and that the secret is not used to encrypt messages. We next handle the case of active adversaries (§4.3), for which we provide sufficient (and rather tight) conditions on protocols for this implication to hold. We establish our transfer result in the applied pi calculus framework (§4.1). Since we do not make any restriction on the use of the replication symbol, protocols with an unbounded number of sessions (as well as bounded number of sessions) can be considered.

4.1 The model

The aim of this section is to briefly introduce the applied pi calculus, and to show how protocols and their secrecy properties can be express within it.

4.1.1 The applied pi calculus

The applied pi calculus [AF01] is a process algebra introduced by M. Abadi and C. Fournet, well-suited for modeling cryptographic protocols, generalising the spi calculus [AG97]. We shortly describe its syntax and semantics. This part is mostly borrowed from [AF01].

We suppose for the moment an arbitrary signature Σ and an arbitrary equational theory $\mathcal{E}$ over Σ. In contrast with the other chapters, we denote here terms by capital letters.

4.1.1.1 Syntax

Processes, also called plain processes, are defined by the grammar:

$$
\begin{array}{lll}
P, Q & := & \text{processes} \\
\quad \mathbf{0} & & \text{null process} \\
\quad P \mid Q & & \text{parallel composition} \\
\quad !P & & \text{replication} \\
\quad \nu n.P & & \text{name restriction} \\
\quad \mathit{if}\ T = T'\ \mathit{then}\ P\ \mathit{else}\ Q & & \text{conditional} \\
\quad u(z).P & & \text{message input} \\
\quad \overline{u}\langle M \rangle.P & & \text{message output}
\end{array}
$$

where n is a name, M, T, T' are terms, and u is a name or a variable. The null process $\mathbf{0}$ does nothing. Parallel composition executes the two processes concurrently. Replication $!P$ creates unboundedly many instances of P. Name restriction $\nu n.P$ builds a new, private name n, binds it in P and then executes P. The conditional *if* $T = T'$ *then* P *else* Q behaves like P or Q depending on the result of the test $T = T'$. If Q is the null process then we use the notation $[T = T'].P$ instead. Finally, the process $u(z).P$ inputs a message on channel u and executes P binding the variable z to the received message, while the process $\overline{u}\langle M \rangle.P$ outputs the message M on channel u and then behaves like P. We may omit P if it is $\mathbf{0}$.

Extended processes are defined by the grammar:

$$
\begin{array}{lll}
A, B & := & \text{extended processes} \\
\quad P & & \text{plain process} \\
\quad A \mid B & & \text{parallel composition} \\
\quad \nu n.A & & \text{name restriction} \\
\quad \nu x.A & & \text{variable restriction} \\
\quad \{^{M}/_{x}\} & & \text{active substitution}
\end{array}
$$

Active substitutions generalise the *let* binding, in the sense that $\nu x.(\{^{M}/_{x}\}|P)$ corresponds to the *let* $x = M$ *in* P standard construction. However, when unrestricted, $\{^{M}/_{x}\}$ behaves like a permanent knowledge, permitting to refer globally to M by means of x. Substitutions $\{^{M_1}/_{x_1}, \ldots, {}^{M_l}/_{x_l}\}$

with $l \geq 0$ are identified with extended processes $\{^{M_1}/_{x_1}\}|\ldots|\{^{M_l}/_{x_l}\}$. In particular, the empty substitution is identified with the null process.

We denote by $\mathrm{fv}(A)$, $\mathrm{bv}(A)$, $\mathrm{fn}(A)$, and $\mathrm{bn}(A)$ the sets of free and bound variables and free and bound names of A respectively. They are defined inductively as usual with $\mathrm{fv}(\{^{M}/_{x}\}) = \mathrm{fv}(M) \cup \{x\}$ and $\mathrm{fn}(\{^{M}/_{x}\}) = \mathrm{names}(M)$ for active substitutions. An extended process is *closed* if it each free variable is defined by an active substitution. For example, $\{^{a}/_{x}\}$ is ground, while $\{^{y}/_{x}\}$ is not.

Extended processes built up from the null process and active substitutions (using the given constructions, that is, parallel composition, restriction and active substitutions) are called *frames*. To every extended process A we associate the frame $\varphi(A)$ obtained by replacing all embedded plain processes with $\mathbf{0}$. For example, if $A = \nu y, k, r.\{^{\mathsf{enc}(m,k,r)}/_{x}, {}^{a}/_{y}\} \mid \overline{c}\langle y\rangle$ then $\varphi(A) = \nu y, k, r.\{^{\mathsf{enc}(m,k,r)}/_{x}, {}^{a}/_{y}\}|\mathbf{0}$.

4.1.1.2 Operational semantics

An *evaluation context* is an extended process with a hole not under a replication, a conditional, an input or an output.

Structural equivalence ($\equiv$) is the smallest equivalence relation on extended processes that is closed by α-conversion of names and variables, by application of evaluation contexts and such that the standard structural rules for the null process, parallel composition and restriction (such as associativity and commutativity of $|$, commutativity and binding behaviour of ν), together with the following three rules hold.

$$\begin{array}{rlr}
\nu x.\{^{M}/_{x}\} \equiv \mathbf{0} & & \textsc{alias}\\
\{^{M}/_{x}\} \mid A \equiv \{^{M}/_{x}\} \mid A\{^{M}/_{x}\} & & \textsc{subst}\\
\{^{M}/_{x}\} \equiv \{^{N}/_{x}\} & \text{if } M =_{\mathcal{E}} N & \textsc{rewrite}
\end{array}$$

If $\widetilde{n}$ represents the (possibly empty) set $\{n_1, \ldots, n_k\}$, we abbreviate by $\nu\widetilde{n}$ the sequence $\nu n_1.\nu n_2 \ldots \nu n_k$. Every closed extended process A can be brought to the form

$$\nu\widetilde{n}.\{^{M_1}/_{x_1}\}|\ldots|\{^{M_l}/_{x_l}\}|P$$

by using structural equivalence, where P is a plain closed process, $l \geq 0$ and $\widetilde{n} \subseteq \cup_i \mathrm{names}(M_i)$. As a consequence, if $A \equiv B$ then $\varphi(A) \equiv \varphi(B)$. Observe also that modulo structural equivalence, frames always take the form $\nu\widetilde{n}.\sigma$ where $\widetilde{n}$ is a finite set of names and σ is an (active) substitution.

Two operational semantics can be considered for this calculus, given by *internal reduction* and by *labeled reduction* respectively. These semantics lead to two equivalence relations between processes: *observational equivalence* (which is standard and not recalled here) and *labeled bisimilarity*. These two bisimilarity relations are in fact equal [AF01], assuming a type system which in particular forbids sending encrypted channel names. We use here the latter since it allows a neater and incremental treatment of the problem we focus on.

Internal reduction is the smallest relation on extended processes which is closed by structural equivalence and application of evaluation contexts, and such that:

$$\begin{array}{lr}
\overline{c}\langle x\rangle.P \mid c(x).Q \ \rightarrow\ P \mid Q & \textsc{comm}\\
\textit{if } T = T' \textit{ then } P \textit{ else } Q \ \rightarrow\ P & \textsc{then}\\
\quad \text{for any ground terms } T \text{ and } T' \text{ such that } T =_{\mathcal{E}} T' & \\
\textit{if } T = T' \textit{ then } P \textit{ else } Q \ \rightarrow\ Q & \textsc{else}\\
\quad \text{for any ground terms } T \text{ and } T' \text{ such that } T \neq_{\mathcal{E}} T' &
\end{array}$$

On the other hand, *labeled reduction* is defined by the following rules:

$$c(x).P \xrightarrow{c(M)} P\{{}^{M}/_{x}\} \ (\ddagger) \quad \text{IN} \qquad \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P \quad \text{OUT-ATOM}$$

$$\frac{A \xrightarrow{\overline{c}\langle u\rangle} A'}{\nu u.A \xrightarrow{\nu u.\overline{c}\langle u\rangle} A'}\ u \neq c \quad \text{OPEN-ATOM} \qquad \frac{A \xrightarrow{\alpha} A'}{\nu u.A \xrightarrow{\alpha} \nu u.A'}\ \begin{matrix}u \text{ does not}\\ \text{occur in } \alpha\end{matrix} \quad \text{SCOPE}$$

$$\frac{A \xrightarrow{\alpha} A'}{A|B \xrightarrow{\alpha} A'|B}\ (\dagger) \quad \text{PAR} \qquad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'} \quad \text{STRUCT}$$

where c is a name and u is a metavariable that ranges over names and variables, and the condition $(\dagger)$ of the rule PAR is $\mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, and the condition $(\ddagger)$ of the rule IN is that M is public w.r.t. $\emptyset$. A term M is said *public* w.r.t. a set of names $\widetilde{n}$ if $\mathrm{names}(M) \cap \widetilde{n} = \emptyset$ and private function symbols do not occur in M (that is, $M \in \mathcal{T}(\mathcal{F}_{\mathsf{pub}}, \mathcal{X}, \mathcal{N} \setminus \widetilde{n})$).

To define labeled bisimilarity we also need an equivalence relation between frames. For a frame $\varphi = \nu\widetilde{n}.\sigma$, we denote $\mathrm{dom}(\varphi) = \mathrm{dom}(\sigma)$ and $\mathrm{ran}(\varphi) = \mathrm{ran}(\sigma)$.

Definition 4.1 *We say that a frame φ* passes the test (U, V) *where U, V are two terms, denoted by $(U = V)\varphi$, if $\varphi = \nu\widetilde{n}.\sigma$, $U\sigma =_{\mathcal{E}} V\sigma$, and $(\mathrm{names}(U) \cup \mathrm{names}(V)) \cap \widetilde{n} = \emptyset$ for some set of names $\widetilde{n}$ and substitution σ.*

Two frames $\varphi = \nu\widetilde{n}.\sigma$ and $\varphi' = \nu\widetilde{m}.\sigma'$ are statically equivalent, *written $\varphi \approx_s \varphi'$, if they pass the same public tests, that is $\mathrm{dom}(\varphi) = \mathrm{dom}(\varphi')$ and for all terms U, V public w.r.t. $\widetilde{n} \cup \widetilde{m}$ such that $(\mathrm{var}(U) \cup \mathrm{var}(V)) \subseteq \mathrm{dom}(\varphi)$, we have $(U = V)\varphi$ if and only if $(U = V)\varphi'$.*

Definition 4.2 Labeled bisimilarity *$(\approx_l)$ is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\,\mathcal{R}\,B$ implies:*

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

4.1.2 Modeling protocols within the applied pi calculus

We work in this chapter with a particular equational theory E, its equations being listed in Figure 4.1. The only functions symbols are those appearing in these equations and $\mathsf{k}(\cdot,\cdot)$. However, the definitions of this section (e.g. those of security properties) are independent of the equational theory in use.

$$\begin{aligned}
&\pi_1(\langle z_1, z_2\rangle) \doteq z_1\\
&\pi_2(\langle z_1, z_2\rangle) \doteq z_2\\
&\mathsf{dec}(\mathsf{enc}(z_1, z_2, z_3), z_2) \doteq z_1\\
&\mathsf{deca}(\mathsf{enca}(z_1, \mathsf{pub}(z_2), z_3), \mathsf{priv}(z_2)) \doteq z_1\\
&\mathsf{check}(z_1, \mathsf{sign}(z_1, \mathsf{priv}(z_2)), \mathsf{pub}(z_2)) \doteq \mathsf{ok}\\
&\mathsf{retrieve}(\mathsf{sign}(z_1, z_2)) \doteq z_1
\end{aligned}$$

Figure 4.1: The equational theory E.

Since protocols are in fact collections of programs executed concurrently, process calculi are good models for protocols as long as they can also handle the cryptographic aspect. The applied pi calculus does this by representing the properties of cryptographic primitives by equations, like the ones in Figure 4.1. Remark that the model of Chapter 1 can also be viewed as a process calculus. However, the applied pi calculus is more expressive since for instance it allows for branching inside a process.

For completeness, we briefly describe next how a process modeling a protocol can be obtained from the model of Chapter 1. Thus, we consider in this chapter roles with equality. Each role is modeled by a process in which the sequence of instructions is represented by a sequence of inputs and outputs, and fresh items are represented by new names under restriction. These processes are first replicated (in order to represent an unbounded number of sessions) and then put in parallel. Since we consider that it is the intruder who starts any role session, we assume that each role process first receives the identities with which it is supposed to communicate, the parameters of roles being thus instantiated. Agent corruption is implicit, in the sense that it is implemented by sending to the environment the private data of corrupted agents.

We exemplify the above discussion by providing a process modeling the Yahalom protocol, which will constitute our running example. We first describe the protocol:

$$\begin{aligned}
A \Rightarrow B &: \ A, N_a \\
B \Rightarrow S &: \ B, \{\!\{A, N_a, N_b\}\!\}_{K_{bs}} \\
S \Rightarrow A &: \ \{\!\{B, K_{ab}, N_a, N_b\}\!\}_{K_{as}}, \{\!\{A, K_{ab}\}\!\}_{K_{bs}} \\
A \Rightarrow B &: \ \{\!\{A, K_{ab}\}\!\}_{K_{bs}}
\end{aligned}$$

In this protocol, two participants A and B wish to establish a shared key K_{ab}. The key is created by a trusted server S which shares the secret keys K_{as} and K_{bs} with A and B respectively. The protocol is modeled by the following process:

$$P_Y = \overline{c}\langle \mathsf{k}(i,s)\rangle \mid (!P_A) \mid (!P_B) \mid (!\nu k.P_S(k)) \mid \nu k_{ab}.P_S(k_{ab})$$

with

$$\begin{aligned}
P_A &= c(z_a).c(z_b).\nu n_a.\overline{c}\langle z_a, n_a\rangle.c(y_a).[z_b = U_b].[n_a = U_{n_a}].\overline{c}\langle \pi_2(y_a)\rangle \\
P_B &= c(z_a).c(z_b).c(y_b).\nu n_b, r_b.\overline{c}\langle z_b, \mathsf{enc}(\langle \pi_1(y_b), \langle \pi_2(y_b), n_b\rangle\rangle, \mathsf{k}(z_b,s), r_b)\rangle. \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad c(y_b').[z_a = \pi_1(\mathsf{dec}(y_b', \mathsf{k}(z_b,s)))] \\
P_S(x) &= c(z_a).c(z_b).c(y_s).[z_a = V_a].[z_b = \pi_1(y_s)].\nu r_s, r_s'. \\
&\qquad \overline{c}\langle\langle \mathsf{enc}(\langle \pi_1(y_s), \langle x, V_n\rangle\rangle, \mathsf{k}(z_a,s), r_s), \mathsf{enc}(\langle V_a, x\rangle, \mathsf{k}(z_b,s), r_s')\rangle\rangle
\end{aligned}$$

where $U_b = \pi_1(\mathsf{dec}(\pi_1(y_a), \mathsf{k}(z_a,s)))$ $\quad U_{n_a} = \pi_1(\pi_2(\pi_2(\mathsf{dec}(\pi_1(y_a), \mathsf{k}(z_a,s)))))$
$V_a = \pi_1(\mathsf{dec}(\pi_2(y_s), \mathsf{k}(z_b,s)))$ $\quad V_n = \pi_2(\mathsf{dec}(\pi_2(y_s), \mathsf{k}(z_b,s)))$.

In order to be able to model the secrecy properties we describe next, we have emphasised a particular key k_{ab} which we will require to remain secret. We have also supposed the existence of a corrupted agent i.

A sample execution of the role of A is given next, where the intruder chooses the concrete agent a and b as participants, obtains the first message sent by A and then sends back a new messages formed by concatenating the identity b with the recently obtained message, and the execution stops since the test does not pass.

$$\begin{aligned}
P_A &\xrightarrow{c(a)}\xrightarrow{c(b)} \nu n_a.\big(\nu z.\{^{\langle a,n_a\rangle}/_z\} \mid \overline{c}\langle z\rangle.c(y_a).[b = U_b].[n_a = U_{n_a}].\overline{c}\langle \pi_2(y_a)\rangle\big) \\
&\xrightarrow{\nu z.\overline{c}\langle z\rangle} \nu n_a.\big(\{^{\langle z_a,n_a\rangle}/_z\} \mid c(y_a).[b = U_b].[n_a = U_{n_a}].\overline{c}\langle \pi_2(y_a)\rangle\big) \\
&\xrightarrow{c(\langle b,z\rangle)} \nu n_a.\big(\{^{\langle z_a,n_a\rangle}/_z\} \mid [b = \pi_1(\mathsf{dec}(b, \mathsf{k}(z_a,s)))].[n_a = U'_{n_a}].\overline{c}\langle z_a, n_a\rangle\big) \to \mathbf{0}
\end{aligned}$$

In what follows, for simplicity and concision, we only consider two honest agents. However, we could extend the processes to the case where the roles of A and B are played by arbitrary agents who may also interact with corrupted identities, and establish a similar result. For example, the process modeling the Yahalom protocol is now:

$$P'_Y = \nu k_{as}, k_{bs}.\,(!P'_A) \mid (!P'_B) \mid (!\nu k.P'_S(k)) \mid \nu k_{ab}.P'_S(k_{ab})$$

with

$$P'_A = \nu n_a.\overline{c}\langle a, n_a\rangle.c(y_a).[b = U'_b].[n_a = U'_{n_a}].\overline{c}\langle \pi_2(y_a)\rangle$$

and similarly for for the other roles (we have mainly eliminated the input of arbitrary parameters and we have replaced z_a, z_b by a, b, and the private terms $\mathsf{k}(z_a, s)$ and $\mathsf{k}(z_b, s)$ by fresh names k_{as} and k_{bs} respectively).

Remark The applied pi calculus relies on a sort system for terms (which is compatible with our sort system of Chapter 1). This sort system is extended to processes and it requires in particular that in input and output constructions $u(x)$ and $\overline{u}\langle N\rangle$, u has sort $\mathsf{Channel}(\tau)$ while x and N have sort τ. Thus, names and variables used for specifying the protocol (having basic sorts) cannot be used as channels.

We give next some notations and lemmas useful in the sequel.

Let $\mathcal{M}_o(P)$ be the set of *outputs* of P, that is the set of terms m such that $\overline{c}\langle m\rangle$ is a message output construct for some channel name c in P, and let $\mathcal{M}_t(P)$ be the set of *operands of tests* of P, where a *test* is a pair $T = T'$ occurring in a conditional and its *operands* are T and T'. Let $\mathcal{M}(P) = \mathcal{M}_o(P) \cup \mathcal{M}_t(P)$ be the set of *messages* of P.

For the Yahalom protocol the set of outputs and operands of tests are respectively:

$$\begin{aligned}
\mathcal{M}_o(P'_Y) = \;& \{\langle a, n_a\rangle, \pi_2(y_a), \langle b, \mathsf{enc}(\langle \pi_1(y_b), \langle \pi_2(y_b), n_b\rangle\rangle, k_{bs}, r_b)\rangle,\\
& \langle \mathsf{enc}(\langle \pi_1(y_s), \langle x, V'_n\rangle\rangle, k_{as}, r_s), \mathsf{enc}(\langle V'_a, x\rangle, k_{bs}, r'_s)\rangle\} \text{ and}\\
\mathcal{M}_t(P'_Y) = \;& \{b, U'_b, n_a, U'_{n_a}, a, \pi_1(\mathsf{dec}(y'_b, k_{bs})), V'_a, \pi_1(y_s)\}.
\end{aligned}$$

where $\quad U'_b = \pi_1(\mathsf{dec}(\pi_1(y_a), k_{as})) \qquad U'_{n_a} = \pi_1(\pi_2(\pi_2(\mathsf{dec}(\pi_1(y_a), k_{as}))))$
$\qquad\quad V'_a = \pi_1(\mathsf{dec}(\pi_2(y_s), k_{bs})) \qquad V'_n = \pi_2(\mathsf{dec}(\pi_2(y_s), k_{bs}))$.

We write $A \Rightarrow B$ if $A \to B$ or $A \xrightarrow{\alpha} B$ for some α.

Definition 4.3 (Valid frame) *A frame φ is* valid *w.r.t. a process P if there is A such that $P \Rightarrow^* A$ and $\varphi \equiv \varphi(A)$.*

The following lemma intuitively states that any message contained in a valid frame is an output instantiated by messages deduced from previous sent messages.

Lemma 4.4 *Let P be a closed plain process, and A be a closed extended process such that $P \Rightarrow^* A$. There are $l \geq 0$, $\widetilde{n} \subseteq \mathrm{bn}(P)$, and*

- *ground substitutions $\sigma_1, \ldots, \sigma_l$ with $\sigma_i = \sigma_{i-1} \uplus \{^{M_i\theta_i\sigma_{i-1}}/_{y_i}\}$, where σ_0 is the empty substitution, and for all $1 \leq i \leq l$, M_i is an output in P, and θ_i is a substitution public w.r.t. $\widetilde{n}$,*

- *an extended process $B = \nu\widetilde{n}.\sigma_l|P_B$, such that $A \equiv B$, where P_B is some plain process,*

- *a substitution θ public w.r.t. $\widetilde{n}$,* *such that for every operand of a test or an output M of P_B there is a message M_0 in P (an operand of a test or an output respectively), with $M = M_0\theta\sigma_l$.*

Proof We provide an inductive and constructive proof. We reason by induction on the number of reductions in $P \Rightarrow^* A$. Intuitively, B is obtained by applying the SUBST rule (from left to right) as much as possible until there are no variables left in the plain process.

The base case is evident.

Assume that $P \Rightarrow^l A_k$ and that there are l, B_l and θ as in the statement of the lemma. Suppose that $A_l \Rightarrow A_{l+1}$ and consider the reduction rule that was used:

- If it is an internal reduction then, since static equivalence is closed by structural equivalence and by internal reduction (see Lemma 1 in [AF01]), it is sufficient to consider as searched values the same as for A_l.

- If it is a labeled reduction then we prove the following property: $\alpha \neq \overline{c}\langle x\rangle$ (for any a and x) and there is an extended process $B_{l+1} = \varphi(B_{l+1})|P_{l+1}$ such that $B_{l+1} \equiv A_{l+1}$ and

  - if $\alpha = \nu x.\overline{c}\langle x\rangle$ then $P_{l+1} = P_l$ and $\varphi(B_{l+1}) = \nu\widetilde{n}.\sigma_{k+1}$, where $\sigma_{k+1} = \sigma_k \uplus \{^{M_l}/_x\}$ and M_l is an output in P_l.
  - if $\alpha = c(M)$ then $\varphi(B_{l+1}) = \varphi(B_l)$ and for every message (an operand of a test or an output) M_{l+1} in P_{l+1} there is a message (an operand of a test or an output, respectively) M_l in P_l, such that $M_{l+1} = M_l\theta'\sigma_k$, for some substitution θ' public w.r.t. $\nu\widetilde{n}$.
  - if $\alpha = \overline{c}\langle n\rangle$ or $\alpha = \nu n.\overline{c}\langle n\rangle$ then $P_{l+1} = P_l$, and $\varphi(B_{l+1}) = \varphi(B_l)$ or $\varphi(B_{l+1}) = \nu\{\widetilde{n}\}\backslash\{n\}.\sigma_k$, respectively.

  It is easy to see that this property is sufficient to prove the inductive step.

  The property can be verified, by showing, using induction on the shape of the derivation tree, that for any extended processes A', A'', B' such that $A' \xrightarrow{\alpha} A''$, $A' \equiv B'$, $B' = \nu\widetilde{n}.\sigma|Q$ there is B'' such that $A'' \equiv B''$ and $B' = \nu\widetilde{n}'.\sigma'|Q'$ where

  - if $\alpha = c(M)$ then $\widetilde{n}' = \widetilde{n}$, $\sigma' = \sigma$ and $N'' = N'\{^M/_x\}$ for each term N'' of B'' where N' is the corresponding term in B' and $c(x)$ is an input in B';
  - if $\alpha = \nu x.\overline{c}\langle x\rangle$ then $Q' = Q$, $\widetilde{n}' = \widetilde{n}$, and $\sigma' = \sigma \uplus \{^M/_x\}$ where $\overline{c}\langle M\rangle$ is an input in B';
  - if $\alpha = \overline{c}\langle x\rangle$, $\alpha = \overline{c}\langle n\rangle$ or $\alpha = \nu n.\overline{c}\langle n\rangle$ then $\widetilde{n}' = \widetilde{n}$ for the first two cases, and $\{\widetilde{n}'\} = \{\widetilde{n}\}\backslash\{n\}$ for the third one, $\sigma' = \sigma$ and $Q' = Q$.

■

Note that B is unique up to the structural rules different from ALIAS, SUBST and REWRITE. We say that $\varphi(B)$ is the *standard frame* w.r.t. A.

We say that a frame $\varphi = \nu\widetilde{n}.\sigma$ is *ground* is σ is ground. Remark that if a frame φ is valid w.r.t. some closed process P then there is a ground frame $\varphi' \equiv \varphi$.

4.1.3 Secrecy properties

4.1.3.1 Passive case

A passive adversary only eavesdrops the communication, and thus he knows the messages sent on the network and also in which order they were sent. As we have already seen, this information is represented in the applied pi calculus by frames. Lemma 4.4 assures that these frames can always be written as $\nu\widetilde{n}.\sigma$ with σ a ground substitution. Thus, in the passive case, we always suppose that frames are ground.

The names in $\widetilde{n}$ are said to be *restricted* in φ. Intuitively, these names are *a priori* unknown to the intruder. The names outside $\widetilde{n}$ are said to be *free* in φ. The set of free names occurring in φ is denoted $\mathrm{fn}(\varphi)$. A term M is said *public* w.r.t. a frame $\nu\widetilde{n}.\sigma$ (or w.r.t. a set of names $\widetilde{n}$) if $\mathrm{names}(M) \cap \widetilde{n} = \emptyset$ and private function symbols do not occur in M (that is, $M \in \mathcal{T}(\mathcal{F}_{\mathsf{pub}}, \mathcal{X}, \mathcal{N} \setminus \widetilde{n})$). The frame or the set of names might be omitted when it is clear from the context, and simply say that a term is public.

In the sequel, we assume that the secret is a term (usually a name denoted by s) of some basic sort, thus not a channel name.

Simple secrecy As we have seen, the intruder knowledge is represented by ground frames. Also, in this chapter, we suppose that all names which are not explicitly restricted (with regard to some process or frame) are available to the intruder. Thus, we define the deducibility relation between ground frames and terms as follows:

$$\varphi \vdash_{\mathcal{E}} M \quad \overset{\text{def}}{\Longleftrightarrow} \quad \mathrm{ran}(\sigma) \cup (\mathcal{N} \setminus \widetilde{n}) \vdash_{\mathcal{I}(\mathcal{E})} M$$

with $\vdash_{\mathcal{I}} (\mathcal{E})$ given by Definition 1.9 (page 39). We drop the subscript $\mathcal{E}$ when it is clear from the context.

A message is usually said secret if it is not in the intruder's knowledge, that is if it not deducible from the messages sent on the network.

Definition 4.5 (Simple secrecy) *We say that a term M is a* simple secret *in φ if $\varphi \not\vdash M$.*

We will often use another characterisation of deducible terms.

Proposition 4.6 *Let $\varphi = \nu\widetilde{n}.\sigma$ be a frame and M be a term. $\varphi \vdash M$ if and only if there exists a public term T w.r.t. φ such that $T\sigma =_{\mathcal{E}} M$.*

This is easily proved by induction on the length of the deducibility proof. It is in fact equivalent with Lemma 1.10.

Exemple 4.7 *The terms k and $\langle k, k'\rangle$ are deducible from the frame $\nu k, k', r.\{^{\mathsf{enc}(k,k',r)}/_{x}, {}^{k'}/_{y}\}$. The "recipes" guaranteed by the previous proposition are $\mathsf{dec}(x, y)$ and $\langle \mathsf{dec}(x, y), y\rangle$ respectively.*

Strong secrecy Deducibility does not always suffice to express all the abilities of an intruder. Some abilities are better captured by static equivalence.

Exemple 4.8 *Let $\sigma_1 = \{^{\mathsf{enc}(n_1,k,r_1)}/_{x}\}$, $\sigma_2 = \{^{\mathsf{enc}(n_2,k,r_2)}/_{x}\}$, $\sigma' = \{^{\langle n_1,n_2\rangle}/_{y}\}$, $\sigma'' = \{^{k}/_{z}\}$, and $\widetilde{n} = \{k, n_1, n_2, r_1\}$. Then the frames $\nu\widetilde{n}.(\sigma_1 \uplus \sigma')$ and $\nu\widetilde{n}.(\sigma_2 \uplus \sigma')$ are statically equivalent, and so are the frames $\nu\widetilde{n}.(\sigma_1 \uplus \sigma'')$ and $\nu\widetilde{n}.(\sigma_2 \uplus \sigma'')$. However, the frames $\varphi_1 = \nu\widetilde{n}.(\sigma_1 \uplus \sigma' \uplus \sigma'')$ and $\varphi_2 = \nu\widetilde{n}.(\sigma_2 \uplus \sigma' \uplus \sigma'')$ are not, since $(\mathsf{dec}(x, z) = \pi_1(y))\varphi_1$ but $(\mathsf{dec}(x, z) \neq \pi_1(y))\varphi_2$.*

Note that the set of deducible messages is the same for all pairs of frames. However, an attacker is able to detect that in the frames φ_1 and φ_2 the first message (i.e. $x\sigma_1$ and $x\sigma_2$ respectively) corresponds to distinct nonces. In particular, the attacker is able to distinguish the two "worlds" represented by φ_1 and φ_2.

Let $\varphi = \nu\widetilde{n}.\sigma$ be a frame and $\mathsf{s} \in \widetilde{n}$ a restricted name in φ. Let M be a term such that $\mathrm{names}(M) \cap \widetilde{n} = \emptyset$. We denote by $\varphi[{}^{M}/_{\mathsf{s}}]$ the frame $\nu\widetilde{n}.\sigma[{}^{M}/_{\mathsf{s}}]$ obtained by instantiating s with M in each term of the substitution σ. For simplicity we may omit s and write $\varphi[M]$ instead of $\varphi[{}^{M}/_{\mathsf{s}}]$.

Definition 4.9 (Strong secrecy) *We say that* s *is a* strong secret *in* φ *if for any closed terms* M, M' *public w.r.t.* φ*, we have* $\varphi[{}^{M}/_{\mathsf{s}}] \approx_s \varphi[{}^{M'}/_{\mathsf{s}}]$.

In other words, s is a strong secret if the intruder cannot distinguish the frames obtained by instantiating the secret s by two terms of its choice.

4.1.3.2 Active case

Given an extended process A we denote by $A[{}^{M}/_{s}]$ the extended process obtained from A by replacing each occurrence of the name s (except the name restrictions νs) with M.

Definition 4.10 (Simple and strong secrecy) *Let* P *be a closed plain process and* s *a bound name of* P*.*

We say that s *is a* simple secret *in* P *if for every ground valid frame* φ *w.r.t.* P*,* $\varphi \not\vdash \mathsf{s}$.

We say that s *is a* strong secret *if for any closed terms* M, M' *public w.r.t.* $\mathrm{bn}(P)$*,* $P[{}^{M}/_{\mathsf{s}}] \approx_l P[{}^{M'}/_{\mathsf{s}}]$.

Examples will be provided in Section 4.3.

4.2 Passive case

4.2.1 Simple secrecy implies strong secrecy

Simple secrecy is usually weaker than strong secrecy! We first exhibit some examples of frames that preserves simple secrecy but not strong secrecy. They all rely on different properties.

Probabilistic encryption. The frame $\psi_1 = \nu\mathsf{s}, k, r.\{{}^{\mathsf{enc}(\mathsf{s},k,r)}/_{x}, {}^{\mathsf{enc}(n,k,r)}/_{y}\}$ does not preserve the strong secrecy of s. Indeed, $\psi_1[n] \not\approx_s \psi_1[n']$ since $(x = y)\, \psi_1[n]$ but $(x \neq y)\, \psi_1[n']$. This would not happen if each encryption used a distinct randomness, that is if the encryption was probabilistic.

Key position. The frame $\psi_2 = \nu\mathsf{s}, n.\{{}^{\mathsf{enc}(\langle n,n'\rangle,\mathsf{s},r)}/_{x}\}$ does not preserve the strong secrecy of s. Indeed, $\psi_2[k] \not\approx_s \psi_2[k']$ since $(\pi_2(\mathsf{dec}(x,k)) = n')\, \psi_2[k]$ but $(\pi_2(\mathsf{dec}(x,k)) \neq n')\, \psi_2[k']$. If s occurs in key position in some ciphertext, the intruder may try to decrypt the ciphertext since s is replaced by public terms and check for some redundancy. It may occur that the encrypted message does not contain any verifiable part. In that case, the frame may preserve strong secrecy. It is for example the case for the frame $\nu n.\{{}^{\mathsf{enc}(n,\mathsf{s},r)}/_{x}\}$. Such cases are however quite rare in practice.

No destructors. The frame $\psi_3 = \nu\mathsf{s}.\{{}^{\pi_1(\mathsf{s})}/_{x}\}$ does not preserve the strong secrecy of s simply because $[x = k]$ is true for $\psi_3[\langle k, k'\rangle]$ while not for $\psi_3[k]$.

Retrieve rule. The $\mathsf{retrieve}(\mathsf{sign}(z_1, z_2)) = z_1$ equation may seem arbitrary since not all signature schemes enable to get the signed message out of a signature. It is actually crucial for our result. For example, the frame $\psi_4 = \nu\mathsf{s}.\{{}^{\mathsf{sign}(\mathsf{s},\mathsf{priv}(a))}/_{x}, {}^{\mathsf{pub}(a)}/_{y}\}$ does not preserve the strong secrecy of s because $[\mathsf{check}(n, x, y) = \mathsf{ok}]$ passes for $\psi_4[n]$ but not for $\psi_4[n']$. However, because of the $\mathsf{retrieve}$ equation, the frame neither preserves the simple secrecy of s.

In the first three cases, the frames preserve the simple secrecy of s, that is $\psi_i \not\vdash \mathsf{s}$, for $1 \leq i \leq 3$. In the fourth case, we would also have $\psi_4 \not\vdash \mathsf{s}$ without the $\mathsf{retrieve}$ equation.

We define agent encryptions as encryptions which use "true" randomness, that is fresh names. Note that in the passive case all encryptions are produced by agents and not by the intruder. Encryption (as a primitive) is probabilistic if each (application of) encryption uses a distinct randomness. Next, we define these notions formally.

We say that an occurrence q_{enc} of an encryption in a term U is an *agent encryption* w.r.t. a set of names $\widetilde{n}$ if $U|_{q_{\mathsf{enc}}\cdot 3} \in \widetilde{n}$. We say that an occurrence q_{enc} of an encryption in a term U is a *probabilistic encryption* w.r.t. a set of terms S if no distinct term shares the same randomness, that is, for any term $V \in S$ and position p such that $V|_p = U|_{q_{\mathsf{enc}}\cdot 3}$ we have that $p = q \cdot 3$ for some q and $V|_q = U|_{q_{\mathsf{enc}}}$.

The previous examples lead us to the following definition.

Definition 4.11 (Well-formed frame) *A frame $\varphi = \nu\widetilde{n}.\sigma$ is* well-formed *w.r.t. some name* s *if*

1. *any encryption in σ is an agent encryption w.r.t. $\widetilde{n}\backslash\{\mathsf{s}\}$ and a probabilistic encryption w.r.t. the set of terms of σ;*
2. s *is not part of a key or a randomness, i.e. for all* $\mathsf{enc}(M, K, R)$, $\mathsf{enca}(M', K', R')$, $\mathsf{sign}(U, V)$, $\mathsf{pub}(W)$, $\mathsf{priv}(W')$ *subterms of φ,* $\mathsf{s} \notin \mathrm{names}(K, K', V, W, W', R, R')$*;*
3. *φ does not contain destructor symbols.*

For well-formed frames, simple secrecy is actually equivalent to strong secrecy.

Theorem 4.12 *Let φ be a well-formed frame w.r.t.* s*, where* s *is a restricted name in φ.*

$$\varphi \nvdash \mathsf{s} \quad \text{if and only if} \quad \varphi[{}^{M}/_{\mathsf{s}}] \approx_s \varphi[{}^{M'}/_{\mathsf{s}}]$$

for all M, M' closed public terms w.r.t. φ.

Proof Let $\varphi = \nu\widetilde{n}.\sigma$ be a well-formed frame w.r.t. s. If $\varphi \vdash \mathsf{s}$, this trivially implies that s is not a strong secret. Indeed, there exists a public term T w.r.t. φ such that $T\sigma =_E \mathsf{s}$, by Proposition 4.6. Let n_1, n_2 be fresh names such that $n_1, n_2 \notin \widetilde{n}$ and $n_1, n_2 \notin \mathrm{fn}(\varphi)$. Since $T\sigma[{}^{n_1}/_{\mathsf{s}}] =_E n_1$ the frames $\varphi[{}^{n_1}/_{\mathsf{s}}]$ and $\varphi[{}^{n_2}/_{\mathsf{s}}]$ are distinguishable by the test $[T = n_1]$.

We assume now that $\varphi \nvdash \mathsf{s}$. We first show that any syntactic equality satisfied by the frame $\varphi[{}^{M}/_{\mathsf{s}}]$ is already satisfied by φ.

Lemma 4.13 *Let $\varphi = \nu\widetilde{n}.\sigma$ be a well-formed frame w.r.t.* $\mathsf{s} \in \widetilde{n}$ *such that* $\varphi \nvdash \mathsf{s}$*. Let U, V and M be public terms w.r.t. φ, with* $\mathrm{var}(U), \mathrm{var}(V) \subseteq \mathrm{dom}(\sigma)$ *and M ground. Then* $U\sigma[{}^{M}/_{\mathsf{s}}] = V\sigma[{}^{M}/_{\mathsf{s}}]$ *implies $U\sigma = V\sigma$.*

This lemma is proved in Section 4.2.2.

The key lemma is that any reduction that applies to a deducible term U where s is replaced by some M, directly applies to U.

Lemma 4.14 *Let $\varphi = \nu\widetilde{n}.\sigma$ be a well-formed frame w.r.t.* $\mathsf{s} \in \widetilde{n}$ *such that* $\varphi \nvdash \mathsf{s}$*. Let U be a term with* $\mathrm{var}(U) \subseteq \mathrm{dom}(\varphi)$ *and M be a closed term in normal form such that U and M are public w.r.t. φ. If $U\sigma[{}^{M}/_{\mathsf{s}}] \to V$, for some term V, then there exists a frame $\varphi' = \nu\widetilde{n}.\sigma'$ well-formed w.r.t.* s

- *extending φ, that is $x\sigma' = x\sigma$ for all $x \in \mathrm{dom}(\sigma)$,*
- *preserving deducible terms: $\varphi \vdash W$ if and only if $\varphi' \vdash W$,*
- *and such that $V = V'\sigma'[^M/_{\mathsf{s}}]$ and $U\sigma \to V'\sigma'$ for some V' public w.r.t. φ'.*

This lemma (proved in Section 4.2.2) allows us to conclude the proof of Theorem 4.12. Fix arbitrarily two public closed terms M, M'. We can assume w.l.o.g. that M and M' are in normal form. Let $U \neq V$ be two public terms such that $\mathrm{var}(U), \mathrm{var}(V) \subseteq \mathrm{dom}(\varphi)$ and $U\sigma[^M/_{\mathsf{s}}] =_E V\sigma[^M/_{\mathsf{s}}]$. Then there are $U_1, \ldots, U_k$ and $V_1, \ldots, V_l$ such that $U\sigma[^M/_{\mathsf{s}}] \to U_1 \to \ldots \to U_k$, $V\sigma[^M/_{\mathsf{s}}] \to V_1 \to \ldots \to V_l$, $U_k = U\sigma[^M/_{\mathsf{s}}]\!\downarrow$, $V_l = V\sigma[^M/_{\mathsf{s}}]\!\downarrow$ and $U_k = V_l$.

Applying repeatedly Lemma 4.14 we obtain that there exist public terms $U'_1, \ldots, U'_k$ and $V'_1, \ldots, V'_l$ and well-formed frames $\varphi_i = \nu\widetilde{n}.\sigma_i$, for $i \in \{1, \ldots, k\}$ and $\psi_j = \nu\widetilde{n}.\theta_j$, for $j \in \{1, \ldots, l\}$ (as in the lemma) such that $U_i = U'_i\sigma_i[^M/_{\mathsf{s}}]$, $U\sigma \to U'_1\sigma_1$, $U'_i\sigma_i \to U'_{i+1}\sigma_{i+1}$, $V_j = V'_j\theta_j[^M/_{\mathsf{s}}]$, $V\sigma \to V'_1\theta_1$ and $V'_j\theta_j \to V'_{j+1}\theta_{j+1}$.

The substitution σ_k extends σ, which means that $\sigma_k = \sigma \uplus \sigma'_k$ with $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma'_k) = \emptyset$. Similarly, $\theta_l = \sigma \uplus \theta'_l$ with $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\theta'_l) = \emptyset$. By possibly renaming the variables of θ'_l and of the V'_j, we can assume that $\mathrm{dom}(\sigma'_k) \cap \mathrm{dom}(\theta'_l) = \emptyset$. We consider $\varphi' = \nu\widetilde{n}.\sigma'$ where $\sigma' = \sigma \uplus \sigma'_k \uplus \theta'_l$. Since only subterms of φ have been added to φ', it is easy to verify that φ' is still a well-formed frame and for every term W we have that $\varphi \vdash W$ if and only if $\varphi' \vdash W$. In particular $\varphi' \nvdash \mathsf{s}$.

By construction we have that $U'_k\sigma_k[^M/_{\mathsf{s}}] = V'_l\theta_l[^M/_{\mathsf{s}}]$. Then, by Lemma 4.13, we deduce that $U'_k\sigma_k = V'_l\theta_l$ that is $U\sigma =_E V\sigma$. By stability of substitution of names, we have $U\sigma[^{M'}/_{\mathsf{s}}] =_E V\sigma[^{M'}/_{\mathsf{s}}]$. We deduce that $\varphi[^M/_{\mathsf{s}}] \approx_s \varphi[^{M'}/_{\mathsf{s}}]$. ∎

4.2.2 Generalisation of well-formed frames

In the active case, we need a more general definition for well-formed frames and for the corresponding lemmas. In particular, we need to consider frames with destructor symbols. Thus we provide here the definition of *extended well-formed* frames, show that well-formed frames are special cases of extended well-formed (when the frames preserve simple secrecy), and then prove analogous lemmas for extended well-formed frames.

In the sequel, especially in the proofs, we often assume a tree visualisation of terms with the root (i.e. the head symbol) at the top, and we thus use notions as "above", "below", "lowest", etc. when talking about occurrences in terms. For example, an occurrence p is above an occurrence q if $p \leq q$. Moreover, we may say that a term V is "in" a term U if V is a subterm of U.

We say that there is an encryption *plaintext-above* a subterm T of a term U at position q_T if there is a position $q < q_T$ such that $U|_q$ is a ciphertext (that is, $\mathrm{head}(U|_q) \in \{\mathsf{enc}, \mathsf{enca}\}$), and T occurs in the plaintext subterm of the encrypted term (that is, $q \cdot 1 \leq q_T$).

Definition 4.15 (Extended well-formed frame) *We say that a frame $\varphi = \nu\widetilde{n}.\sigma$ is an* extended well-formed *w.r.t.* s *if*

1. *all the terms of σ are in normal form,*
2. *any agent encryption w.r.t. $\widetilde{n}$ in σ is a probabilistic encryption w.r.t. $\mathrm{ran}(\sigma)$,*
3. *for every occurrence q_{s} of s in $y\sigma$ with $y \in \mathrm{dom}(\sigma)$, there exists an agent encryption (say q_{enc}) w.r.t. $\widetilde{n}\backslash\{\mathsf{s}\}$ plaintext-above s,*

4. *the lowest agent encryption q_0 plaintext-above* s *satisfies* $\text{head}(y\sigma|_q) \in \{\langle\rangle, \mathsf{sign}\}$, *for all positions q with $q_0 < q < q_{\mathsf{s}}$.*

This definition ensures in particular that there is no destructor directly above s.

Exemple 4.16 *The frame $\varphi = \nu\mathsf{s}, k, n.\{^{\pi_1(\mathsf{enc}(a,\mathsf{enc}(\langle b,\mathsf{s}\rangle,k,n)),n'')}/_x, {}^{\mathsf{enc}(a,k',n')}/_y, {}^{\mathsf{enc}(b,k',n')}/_z\}$ is extended well-formed, while the frames $\varphi_2 = \nu n.\{^{\mathsf{enc}(a,k,n)}/_y, {}^{\mathsf{enc}(b,k,n)}/_z\}$, $\varphi_3 = \nu n.\{^{\mathsf{enc}(a,\mathsf{s},n)}/_x\}$, and $\varphi_4 = \nu\mathsf{s}, k, n.\{^{\mathsf{enc}(\pi_1(\mathsf{s}),k,n)}/_x\}$ are not, each frame φ_i contradicting condition i. of the Definition 4.15 (i.e. φ_1 contradicts condition 1. and so on).*

We first start by a preliminary lemma which states that in a well-formed frame w.r.t. s, either every occurrence of s is under some encryption or s is deducible.

Lemma 4.17 *Let $\varphi = \nu\widetilde{n}.\sigma$ be a well-formed frame w.r.t. $\mathsf{s} \in \widetilde{n}$ such that $\varphi \nvdash \mathsf{s}$. For any $y \in \text{dom}(\sigma)$ and for any occurrence q_{s} of s in $y\sigma$ there is an encryption plaintext-above q_{s} in $y\sigma$.*

Proof Assume by contradiction that there is an occurrence q_{s} of s in $y\sigma$ such that there is no encryption plaintext-above s. Then, from conditions 2 and 3 of the definition of well-formed frames, we have that there are only pairs and signatures as function symbols above s. It follows that s is deducible (by applying the projections and the $\mathsf{retrieve}$ equations), which contradicts the hypothesis.

Thus, there exists a position $q < q_{\mathsf{s}}$ such that $y\sigma|_q$ is an encryption. By condition 2 of the definition of well-formed frames, s must occur in the plaintext part of the encryption, that is $q \cdot 1 \leq q_{\mathsf{s}}$. ■

Lemma 4.18 *Let $\varphi = \nu\widetilde{n}.\sigma$ be a frame and s a restricted name in φ such that $\varphi \nvdash \mathsf{s}$. If φ is a well-formed frame w.r.t. s then it is an extended well-formed frame w.r.t. s.*

Proof Since there are no destructor symbols in φ all terms are in normal form. Since any encryption in σ is probabilistic it will be a fortiori the case for agent encryptions.

Consider an occurrence q_{s} of s in $y\sigma$ with $y \in \text{dom}(\sigma)$. From Lemma 4.17 we have that there is at least an encryption plaintext-above s in $y\sigma$. Consider the lowest one. Then condition 1 of the definition of well-formed frames says that this encryption is an agent encryption. Conditions 2 and 3 impose that the only function symbols in between may be $\langle\rangle$ and sign. ■

The following lemma states that if in two distinct terms the secret is protected by agent probabilistic encryptions then by replacing the secret with any term we cannot obtain two syntactically equal terms.

Lemma 4.19 *Let $\widetilde{n}$ be a set of names and s be a name, $\mathsf{s} \in \widetilde{n}$. Let M be a ground public term w.r.t. $\widetilde{n}$ and U, V be two terms such that for any occurrence q_{s} of s (in U or V) there is an encryption q_{enc} (in U or V respectively) with $q_{\mathsf{enc}} \cdot 1 \leq q_{\mathsf{s}}$ such that q_{enc} is an agent encryption w.r.t. $\widetilde{n}\backslash\{\mathsf{s}\}$ and q_{enc} is a probabilistic encryption w.r.t. $\{U, V\}$. Then $U[^M/_{\mathsf{s}}] = V[^M/_{\mathsf{s}}]$ implies $U = V$.*

Proof Suppose that $U[^M/_{\mathsf{s}}] = V[^M/_{\mathsf{s}}]$ and $U \neq V$. Then there is an occurrence q_{s} of s, say in U, such that $V|_{q_{\mathsf{s}}} \neq \mathsf{s}$. Consider an agent probabilistic encryption q_{enc} with $q_{\mathsf{enc}} \cdot 1 \leq q_{\mathsf{s}}$ as in the lemma. We have $U|_{q_{\mathsf{enc}} \cdot 3} \in \widetilde{n}\backslash\{\mathsf{s}\}$. It follows that $V[^M/_{\mathsf{s}}]|_{q_{\mathsf{enc}} \cdot 3} \in \widetilde{n}\backslash\{\mathsf{s}\}$. Since M is public this implies that $q_{\mathsf{enc}} \cdot 3$ is a position in V. And since q_{enc} is a probabilistic encryption and $U|_{q_{\mathsf{enc}} \cdot 3} = V|_{q_{\mathsf{enc}} \cdot 3}$ it follows that $U|_{q_{\mathsf{enc}}} = V|_{q_{\mathsf{enc}}}$. Hence $U|_{q_{\mathsf{s}}} = V|_{q_{\mathsf{s}}}$ which represents a contradiction with $V|_{q_{\mathsf{s}}} \neq \mathsf{s}$. ■

Corollary 4.20 *Let $\varphi = \nu\widetilde{n}.\sigma$ be an extended well-formed frame w.r.t. $\mathsf{s} \in \widetilde{n}$ such that $\varphi \nvdash \mathsf{s}$. Let U, V and M be public terms w.r.t. φ, with $\mathrm{var}(U), \mathrm{var}(V) \subseteq \mathrm{dom}(\sigma)$ and M ground. Let W, W' be subterms of terms in $\mathrm{ran}(\sigma)$ such that for every occurrence q_{s} of s in W (or W') there is an occurrence of an encryption q_{enc} in W (or W' respectively) with $q_{\mathsf{enc}} < q_{\mathsf{s}}$. Then*

1. $U\sigma[{}^{M}/_{\mathsf{s}}] = V\sigma[{}^{M}/_{\mathsf{s}}]$ *implies* $U\sigma = V\sigma$;
2. $U\sigma[{}^{M}/_{\mathsf{s}}] = W[{}^{M}/_{\mathsf{s}}]$ *implies* $U\sigma = W$;
3. $W[{}^{M}/_{\mathsf{s}}] = W'[{}^{M}/_{\mathsf{s}}]$ *implies* $W = W'$.

Proof We prove below that in $U\sigma$ and in W for each occurrence q_{s} of s there is an encryption q'_{enc} (in $y\sigma$ for some $y \in \mathrm{var}(U)$, and in W respectively) with $q'_{\mathsf{enc}} \cdot 1 \leq q_{\mathsf{s}}$ such that q'_{enc} is an agent encryption w.r.t. $\widetilde{n} \backslash \{\mathsf{s}\}$. Then, by analogy, the same thing holds for $V\sigma$ and W'. Since by condition (2) of extended well-formed frames an agent encryption w.r.t. $\widetilde{n}$ is a probabilistic encryption, it follows that each pair $(U\sigma, V\sigma)$, $(U\sigma, W)$ and (W, W') satisfies the conditions of Lemma 4.19. Then the result follows directly.

Consider an occurrence q_{s} of s in $U\sigma$. Since U is public, there is a variable $y \in \mathrm{var}(U) \subseteq \mathrm{dom}(\sigma)$ and an occurrence p_y of it in U such that $p_y \leq q_{\mathsf{s}}$. From the definition of extended well-formed frames we know that there is an encryption q'_{enc} in $y\sigma$ with $q'_{\mathsf{enc}} \cdot 1 \leq q_{\mathsf{s}}$ which is an agent encryption w.r.t. $\widetilde{n} \backslash \{\mathsf{s}\}$. Hence q'_{enc} satisfies the conditions of Lemma 4.19.

In W for each occurrence q_{s} of s there is an occurrence q_{enc} of an encryption above q_{s}. Then we can consider the lowest occurrence q'_{enc} of an encryption above q_{s} in W. By the definition of extended well-formed frames, the lowest encryption above q_{s} is an agent encryption and is plain-text above q_{s}. Hence q'_{enc} satisfies the conditions of Lemma 4.19. ∎

Lemma 4.13 can now be easily deduced since it is the analogous statement of Point 1 of Corollary 4.20 for well-formed frames (which are extended well-formed frames as we have seen in Lemma 4.18).

The following lemma is the generalisation of Lemma 4.14 for extended well-formed frames.

Lemma 4.21 *Let $\varphi = \nu\widetilde{n}.\sigma$ be an extended well-formed frame w.r.t. $\mathsf{s} \in \widetilde{n}$ such that $\varphi \nvdash \mathsf{s}$. Let U be a term with $\mathrm{var}(U) \subseteq \mathrm{dom}(\varphi)$ and M be a closed term in normal form such that U and M are public w.r.t. φ. If $U\sigma[{}^{M}/_{\mathsf{s}}] \to V$, for some term V, then there exists an extended well-formed frame $\varphi' = \nu\widetilde{n}.\sigma'$ w.r.t. s*

- *extending φ, that is $x\sigma' = x\sigma$ for all $x \in \mathrm{dom}(\sigma)$,*
- *preserving deducible terms: $\varphi \vdash W$ if and only if $\varphi' \vdash W$,*
- *and such that $V = V'\sigma'[{}^{M}/_{\mathsf{s}}]$ and $U\sigma \to V'\sigma'$ for some V' public w.r.t. φ'.*

Proof Let U, V, M be terms with U and M public w.r.t. φ, M being closed and in normal form such that $U\sigma[{}^{M}/_{\mathsf{s}}] \to V$, as in the statement of the lemma. Let $(L \to R) \in \mathcal{R}(E)$ be the rule that was applied in the above reduction and let p be the position at which it was applied, *i.e.* $U\sigma[{}^{M}/_{\mathsf{s}}]|_p = L\theta$. Since M is in normal form, $p \in \mathrm{pos}(U\sigma)$.

Assume that there is a substitution θ_0 such that $U\sigma|_p = L\theta_0$. This will be proved in the Fact below. It follows that $U\sigma$ is reducible. If $p \notin \mathrm{pos}_{\mathrm{nv}}(U)$ then there is a term of $\mathrm{ran}(\sigma)$ which is reducible. This contradicts the fact that φ is an extended-well formed frame (since all terms in such a frame should be in normal form). Hence we have that $p \in \mathrm{pos}_{\mathrm{nv}}(U)$. Let $T = U|_p$. We have $T\sigma[{}^{M}/_{\mathsf{s}}] = L\theta$ and $T\sigma = L\theta_0$.

For our equational theory E, R is either a constant (*i.e.* ok) or a variable. If R is a constant then we take $V' = U[R]_p$ and $\sigma' = \sigma$. It is easy to verify that the conditions of the lemma are satisfied in this case.

Suppose now that R is a variable z_0. Consider the[14] position q of z_0 in L. This position q is also in $L\theta_0$, that is in $T\sigma$. Hence one of the two following possibilities may occur:

1. If $q \in \mathrm{pos}_{\mathrm{nv}}(T)$, that is there is no $y \in \mathrm{dom}(\sigma)$ above z_0, then we consider $V' = U[T|_q]_p$ and $\sigma' = \sigma$. In this case also, it is easy to verify that the conditions of the lemma are satisfied.

2. If $q \notin \mathrm{pos}_{\mathrm{nv}}(T)$, that is there is some $y \in \mathrm{dom}(\sigma)$ above z_0, then we consider $V' = U[y']_p$ and $\sigma' = \sigma \uplus \{R\theta_0/y'\}$, where y' is a new variable (i.e. $y' \notin \mathrm{dom}(\sigma)$). The term V' is clearly public w.r.t. φ'. Since $T\sigma =_E R\theta_0$, $\varphi \vdash R\theta_0$. This shows that $\varphi \vdash W$ if and only if $\varphi' \vdash W$ for any term W by using the cut-elimination lemma (see Lemma 2.14 at page 63)

   We have $V'\sigma' = (U[y']_p)\sigma' = U\sigma'[y'\sigma']_p = U\sigma[R\theta_0]_p$. Hence $U\sigma \rightarrow V'\sigma'$.

   From $T\sigma = L\theta_0$ and $T\sigma[{}^{M}/_{\mathsf{s}}] = L\theta$ we deduce that $z\theta_0[{}^{M}/_{\mathsf{s}}] = z\theta$ for all $z \in \mathrm{var}(L)$, hence $R\theta_0[{}^{M}/_{\mathsf{s}}] = R\theta$. Thus $V'\sigma'[{}^{M}/_{\mathsf{s}}] = (U\sigma[{}^{M}/_{\mathsf{s}}])[R\theta]_p = V$.

   Since there is some $y \in \mathrm{dom}(\varphi)$ above z_0, $R\theta_0 = z_0\theta$ is a subterm of a term of σ. Then $R\theta_0$ is in normal form since all the terms in $\mathrm{ran}(\sigma)$ are in normal form. Also all agent encryptions in φ' are probabilistic. Suppose that there is an occurrence of s in $R\theta_0$ such that there is no encryption plaintext-above it (in $R\theta_0$). In this case we have that all the function symbols above this occurrence in $R\theta_0$ are $\langle\rangle$ or sign. Thus s is deducible from φ' and hence from φ, which represents a contradiction with the hypothesis. Hence there is an encryption plaintext-above any occurrence of s in $R\theta_0$. All this proves that φ' is also an extended well-formed frame.

Fact: Let us now prove that there exists θ_0 such that $U\sigma|_p = L\theta_0$. Assume by contradiction that it is not the case. Then at least one of the following cases occurs:

1. there is a position in L which is not a position in $U\sigma|_p$;

2. there is a variable z in L having at least two occurrences, say at positions p_1, p_2, for which $(U\sigma|_p)|_{p_1} \neq (U\sigma|_p)|_{p_2}$.

Let us examine in detail the two cases:

1. Consider a minimal position q' (w.r.t. the prefix ordering) in L which is not a position in $U\sigma|_p$. Then $q' = q \cdot i$ for some positive integer i, with q a position of $U\sigma|_p$ and there is an s at position q in $U\sigma|_p$ (since such minimal positions in L must be positions in $U\sigma[{}^{M}/_{\mathsf{s}}]|_p$, but not in $U\sigma|_p$). Also $q \neq \epsilon$ (*i.e.* it does not correspond to the head of L) since otherwise M would not be in normal form (because $U\sigma|_p = s$ and $U\sigma[{}^{M}/_{\mathsf{s}}]|_p = M = L\theta$).

   By examining all rules in $\mathcal{R}(E)$, we observe that at least one of the conditions in the definition of extended well-formed frames is not satisfied. For example, if $L \rightarrow R$ is the rule $\pi_1(\langle z_1, z_2\rangle) \rightarrow z_1$ then $q = 1$. Then either $\pi_1(y)$ is the subterm at position p in U and $y\sigma = \mathsf{s}$ (impossible case since s would be deducible), or $\pi_1(\mathsf{s})$ is the subterm at position p in $U\sigma$ and this subterm is also a subterm of a term of σ (again an impossible case because there are no destructors right above s in term of an extended well-formed

[14] For our equational theory there is exactly one occurrence of z_0 in L.

frame). If $L \to R$ is the rule $\mathsf{deca}(\mathsf{enca}(z_1, \mathsf{pub}(z_2), z_3), \mathsf{priv}(z_2)) \to z_1$ then q might be 1 or $1 \cdot 2$. The case $q = 1$ is similar with the previous one. If $q = 1 \cdot 2$ then we have a term in σ having $\mathsf{enca}(W, \mathsf{s})$ as subterm for some W (otherwise s would be deducible). But this again contradicts the definition of extended well-formed frames. The analysis for the other rules is similar.

2. Let $T_1 = (U\sigma|_p)|_{p_1}$ and $T_2 = (U\sigma|_p)|_{p_2}$. We have $T_1 \neq T_2$, but $T_1[{}^{M}/_{\mathsf{s}}] = T_2[{}^{M}/_{\mathsf{s}}]$. Consider an arbitrary position q_{s} of s in T_1. Since U is public, there is a variable $y \in \mathrm{var}(U)$ at position say p_y such that $p_y \leq p \cdot p_1 \cdot q_{\mathsf{s}}$. Consider the lowest agent encryption q_{enc} plaintext-above q_{s} in $U\sigma$. It occurs in $y\sigma$ according to the definition of extended well-formed frames. Suppose that $p \cdot p_1 > q_{\mathsf{enc}}$. The function symbols between q_{enc} and $p \cdot p_1$ must be $\langle\rangle$ or sign. But this doesn't hold for none of rules in $\mathcal{R}(E)$. Hence there is an agent encryption plaintext-above q_{s} in T_1. The same argument applies to T_2. We can thus use Point 3 of Corollary 4.20 to T_1 and T_2 and obtain a contradiction, that is $T_1 = T_2$.

We have seen that the two cases lead to contradictions. So there is θ_0 such that $U\sigma|_p = L\theta_0$. ∎

4.3 Active case

4.3.1 Our hypotheses

In what follows, we assume s to be the desired secret. As in the passive case, destructors above the secret must be forbidden. We also restrict ourself to processes with ground terms in key position. Indeed, consider the process

$$P_1 = \nu\mathsf{s}, k, r, r'.\big(\overline{c}\langle \mathsf{enc}(\mathsf{s}, k, r)\rangle \mid c(z).\overline{c}\langle \mathsf{enc}(a, \mathsf{dec}(z, k), r')\rangle\big).$$

The name s in P_1 is a simple secret but not a strong secret. Indeed,

$$\begin{array}{rl} P_1 & \equiv\ \nu\mathsf{s}, k, r, r'.\big(\nu x.(\{{}^{\mathsf{enc}(\mathsf{s},k,r)}/_{x}\} \mid \overline{c}\langle x\rangle \mid c(z).\overline{c}\langle \mathsf{enc}(a, \mathsf{dec}(z, k), r')\rangle)\big) \\ & \to\ \nu\mathsf{s}, k, r, r'.\big(\{{}^{\mathsf{enc}(\mathsf{s},k,r)}/_{x}\} \mid \overline{z}\langle \mathsf{enc}(a, \mathsf{s}, r')\rangle\big) \qquad\qquad (\textsc{comm}\ \text{rule}) \\ & \equiv\ \nu\mathsf{s}, k, r, r'.\big(\nu y.(\{{}^{\mathsf{enc}(\mathsf{s},k,r)}/_{x}, {}^{\mathsf{enc}(a,\mathsf{s},r')}/_{y}\} \mid \overline{c}\langle y\rangle)\big) \\ & \xrightarrow{\nu y.\overline{c}\langle y\rangle}\ P_1' = \nu\mathsf{s}, k, r, r'.\{{}^{\mathsf{enc}(\mathsf{s},k,r)}/_{x}, {}^{\mathsf{enc}(a,\mathsf{s},r')}/_{y}\} \end{array}$$

and P_1' does not preserve the strong secrecy of s, since the frame $\varphi(P_1')$ does not preserve it. Indeed, using the same idea as for the frame ψ_2 of Section 4.2.1, one distinguishing test would be $[dec(y, k') = a]$ for some public name k'. This test would succeed when s is instantiated by k' but not if s is instantiated by some other value, say k''.

We denote by $\mathsf{enc_g}$ (respectively $\mathsf{dec_g}$) a generic encryption (decryption), that is when using it we refer to both symmetric and asymmetric encryption (decryption)[15].

Without loss of generality with respect to cryptographic protocols, we assume that terms occurring in processes are in normal form and that no destructor appears above constructors. Indeed, terms like $\pi_1(\mathsf{enc_g}(M, K, R))$ are usually not used to specify protocols. We also assume that tests do not contain constructors. Indeed a test $[\langle T_1, T_2\rangle = T']$ can be rewritten as $[T_1 = T_1'].[T_2 = T_2']$ if $T' = \langle T_1', T_2'\rangle$, and $[T_1 = \pi_1(T')].[T_2 = \pi_2(T')]$ if T' does not contain constructors, and will never hold otherwise. Similar rewriting applies for encryption, except for the test

[15]For example, when $\mathsf{enc_g}$ is under universal quantification one would read enc *and* encd, while under existential quantification one would read enc *or* encd for $\mathsf{enc_g}$.

$[\mathsf{enc_g}(T_1, T_2, T_3) = T']$ if T' does not contain constructors. It can be rewritten in $[\mathsf{dec_g}(T', T_2) = T_1]$ but this is not equivalent. However since the randomness of encryption is not known to the agents, explicit tests on the randomness should not occur in general.

This leads us to consider the following class of processes.

Definition 4.22 (Well-formed process) *A process P is* well-formed *w.r.t. a name* s *if it is closed, and:*

1. *the symbol* retrieve *does not occur in $\mathcal{M}(P)$, the symbol* check *does not occur in $\mathcal{M}(P)$ except in head of a test, that is, the* check *symbol can only appear in tests of the form* $[\mathsf{check}(M, N, K) = \mathsf{ok}]$ *where* check *does not appear in M, N, K;*

2. *any encryption in some term of $\mathcal{M}(P)$ is a probabilistic agent encryption w.r.t. $\mathcal{M}(P)$ and* $\mathrm{bn}(P)\backslash\{\mathsf{s}\}$ *respectively;*

3. *for any term $\mathsf{enc_g}(M, K, R)$, $\mathsf{dec_g}(M, K)$ or $\mathsf{sign}(M, K)$ occurring in $\mathcal{M}(P)$, K is a closed term and $\mathsf{s} \notin \mathrm{names}(K)$;*

4. *in $\mathcal{M}(P)$ there are no destructors, nor* pub *or* priv *function symbols above constructors, nor above* s*;*

5. *for any test,*

   - *either each operand of a test $T \in \mathcal{M}_t$ is a name, a constant or has the form*

   $$\pi^1(\mathsf{dec}_1(\ldots \pi^l(\mathsf{dec}_l(\pi^{l+1}(z), K_l)) \ldots, K_1))$$

   with $l \geq 0$, where $\mathsf{dec}_i \in \{\mathsf{dec}, \mathsf{deca}\}$, π^i are words on $\{\pi_1, \pi_2\}$ and z is a variable,

   - *or the test is $[\mathsf{check}(M, N, K) = \mathsf{ok}]$ with K being a closed term, and M and N being of the previously described form.*

Conditionals should not test on s. For example, consider the following process:

$$P_2 = \nu \mathsf{s}, k, r.\big(\overline{c}\langle \mathsf{enc}(\mathsf{s}, k, r)\rangle \mid c(z).[\mathsf{dec}(z, k) = a].\overline{c}\langle \mathsf{ok}\rangle\big)$$

where a is a non restricted name. The name s in P_2 is a simple secret but not a strong secret. Indeed, $P_2 \to \nu \mathsf{s}, k, r.(\{^{\mathsf{enc}(\mathsf{s},k,r)}/_z\} \mid [\mathsf{s} = a].\overline{c}\langle \mathsf{ok}\rangle)$ and the process $P_2[^a/_\mathsf{s}]$ reduces further, while $P_2[^b/_\mathsf{s}]$ does not.

That is why we have to prevent hidden tests on s. Such tests may occur nested in equality tests. For example, let

$$\begin{aligned} P_3 = {} & \nu \mathsf{s}, k, r, r_1, r_2.\big(\overline{c}\langle \mathsf{enc}(\mathsf{s}, k, r)\rangle \mid \overline{c}\langle \mathsf{enc}(\mathsf{enc}(a, k', r_2), k, r_1)\rangle \\ & \qquad \mid c(z).[\mathsf{dec}(\mathsf{dec}(z, k), k') = a].\overline{c}\langle \mathsf{ok}\rangle\big) \quad \to \\ P_3' = {} & \nu \mathsf{s}, k, r, r_1, r_2.\big(\{^{\mathsf{enc}(\mathsf{s},k,r)}/_z\} \mid \overline{c}\langle \mathsf{enc}(\mathsf{enc}(a, k', r_2), k, r_1)\rangle \mid [\mathsf{dec}(\mathsf{s}, k') = a].\overline{c}\langle \mathsf{ok}\rangle\big) \end{aligned}$$

Then $P_3[^{\mathsf{enc}(a,k',r')}/_\mathsf{s}]$ is not equivalent to $P_3[^n/_\mathsf{s}]$, since the process $P_3'[^{\mathsf{enc}(a,k',r')}/_\mathsf{s}]$ emits the message ok while $P_3'[^n/_\mathsf{s}]$ does not. This relies on the fact that the decryption $\mathsf{dec}(z, k)$ allows access to s in the test.

For the remaining of the section we assume that $\mathtt{x}$ and $\mathtt{z}_0$ are new fixed variables. To prevent hidden tests on the secret, we compute an over-approximation of the ciphertexts that may contain the secret, by marking with $\mathtt{x}$ all positions under which the secret may appear in clear.

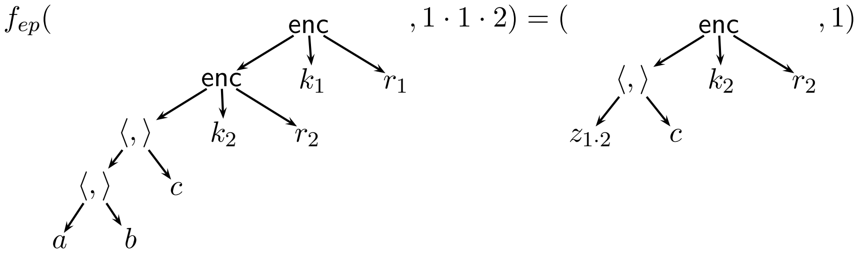
We first introduce a function f_{ep} that extracts the lowest encryption over $\mathtt{s}$ and "cleans up" the pairing and signing functions above $\mathtt{s}$. Formally, we define the partial function

$$f_{ep}\colon \mathcal{T} \times \mathbb{N}_+^* \quad \hookrightarrow \quad \mathcal{T} \times \mathbb{N}_+^*$$

$f_{ep}(U,p) = (V,q)$ where V and q are defined as follows: $q \leq p$ is the position (if it exists) of the lowest encryption on the path p in U. If q does not exist or if p is not a maximal position in U or if $q \cdot 1 \not\leq p$, then $f_{ep}(U,p) = \perp$. Otherwise, V is obtained from $U|_q$ by replacing all arguments of pairs and signatures that are not on the path p with new variables. More precisely, let $V' = U|_q$. If the subterm V' is not of the form $\mathsf{enc_g}(M_1, M_2, M_3)$ or if $p \neq q \cdot 1 \cdot q'$ for some position q' then $f_{ep}(U,p) = \perp$. Otherwise, V is defined by $V = \mathsf{enc_g}(M_1', M_2, M_3)$ with $M_1' = \mathsf{prune}(M_1, q')$ where prune is recursively defined by:

$$\begin{array}{l}
\mathsf{prune}(N,\epsilon) = N \\
\mathsf{prune}(\langle N_1, N_2\rangle, 1\cdot r) = \langle \mathsf{prune}(N_1, r), x_{2\cdot r}\rangle \\
\mathsf{prune}(\langle N_1, N_2\rangle, 2\cdot r) = \langle x_{1\cdot r}, \mathsf{prune}(N_2, r)\rangle \\
\mathsf{prune}(\mathsf{sign}(M,K), 1\cdot r) = \mathsf{sign}(\mathsf{prune}(M), x_{2\cdot r}) \\
\mathsf{prune}(f(N_1,\dots,N_k), r) = f(N_1,\dots,N_k) \quad \text{if } f \text{ is a destructor}
\end{array}$$

and is undefined in all other cases. For example,



The function f_e is the composition of the first projection with f_{ep}. With the function f_e, we can extract from the outputs of a protocol P the set of ciphertexts where $\mathtt{s}$ appears explicitly below the encryption.

$$\mathcal{E}_0(P) = \{ f_e(M[\mathtt{x}]_p, p) \mid M \in \mathcal{M}_o(P) \ \wedge\ M|_p = \mathtt{s} \}.$$

For example, $\mathcal{E}_0(P_Y') = \{\mathsf{enc}(\langle z_{1\cdot 1}, \langle \mathtt{x}, z_2\rangle\rangle, k_{as}, r_s), \mathsf{enc}(\langle z_1, \mathtt{x}\rangle, k_{bs}, r_s')\}$, where P_Y' is the process corresponding to the Yahalom protocol defined in previous section and $\mathtt{s}$ denotes k_{ab}.

However $\mathtt{s}$ may appear in other ciphertexts sent later on during the execution of the protocol after decryptions and encryptions. Thus we also extract from outputs the destructor parts (which may open encryptions). Namely, we define the partial function

$$f_{dp}\colon \mathcal{T} \times \mathbb{N}_+^* \hookrightarrow \mathcal{T} \times \mathbb{N}_+^*$$

$f_{dp}(U,p) = (V,q)$ where V and q are defined as follows: $q \leq p$ is the occurrence of the highest destructor different from check above p (if it exists). Let $r \leq p$ be the occurrence of the lowest decryption above p (if it exists). We have $U|_r = \mathsf{dec_g}(U_1, U_2)$. Then U_1 is replaced by the variable z_0 that is $V = (U[\mathsf{dec_g}(\mathrm{z}_0, U_2)]_r)|_q$. If q or r do not exist then $f_{dp}(U,p) = \perp$.

For example, $f_{dp}(\mathsf{enc}(\pi_1(\mathsf{dec}(\pi_2(y), k_1)), k_2, r_2), 1\cdot 1\cdot 1\cdot 1) = (\pi_1(\mathsf{dec}(\mathrm{z}_0, k_1)), 1)$.

The function f_d is the composition of the first projection with f_{dp}. By applying the function f_d to messages of a well-formed process P we always obtain either terms D of the form[16] $D =$

[16]in this context we simply write $D(T)$ instead of $D[{}^{T}/_{\mathrm{z}_0}]$

$D_1(\dots D_n)$ where $D_i(z_0) = \pi^i(\mathsf{dec_g}(z_0, K_i))$ with $1 \le i \le n$, K_i are ground terms and π^i is a (possibly empty) sequence of projections $\pi_{j_1}(\pi_{j_2}(\dots(\pi_{j_l})\dots))$, or terms $\mathsf{check}(M, D, K)$ where D is of the previously defined form.

With the function f_d, we can extract from the outputs of a protocol P the meaningful destructor part.

$$\mathcal{D}_o(P) = \{f_d(M, p) \mid M \in \mathcal{M}_o(P) \ \wedge \ p \in \mathrm{pos_v}(M)\}.$$

Remember that $\mathrm{pos_v}(M)$ is the set of variable positions.

For example, $\mathcal{D}_o(P'_Y) = \{\pi_2(\mathsf{dec}(z_0, k_{bs})), \pi_1(\mathsf{dec}(z_0, k_{bs}))\}$.

We are now ready to mark (with $\mathtt{x}$) all the positions where the secret might be transmitted (thus tested). We define inductively the sets $\mathcal{E}_i(P)$ as follows. For each element E of $\mathcal{E}_i$ we can show that there is an unique term in normal form denoted by $\overline{E}$ such that $\mathrm{var}(\overline{E}) = \{z_0\}$ and $\overline{E}(E)\!\downarrow = \mathtt{x}$. That is, intuitively, $\overline{E}$ opens E until $\mathtt{x}$. For example, let $E_1 = \mathsf{enc}(\langle z_1, \langle \mathtt{x}, z_2\rangle\rangle, k_{as}, r_s)$, then $\overline{E_1} = \pi_1(\pi_2(\mathsf{dec}(z_0, k_{as})))$. We define

$$\begin{array}{rcl}\overline{\mathcal{E}_i}(P) & = & \{U \mid \exists E \in \mathcal{E}_i(P), U \le_{st} \overline{E} \ \text{ and } \exists q \in \mathrm{pos}(U), \mathrm{head}(U|_q) = \mathsf{dec_g}\},\\ \mathcal{E}_{i+1}(P) & = & \{M'[\mathtt{x}]_q \mid \exists M \in \mathcal{M}_o(P), p \in \mathrm{pos_v}(M) \text{ s.t. } f_{ep}(M,p) = (M', p'),\\ & & \quad f_{dp}(M', p'') = (D, q), p = p' \cdot p'', D = D_1(\dots D_n), \text{ and } D_1 \in \overline{\mathcal{E}}_i(P)\}.\end{array}$$

For example,
$\overline{\mathcal{E}_0}(P'_Y) = \{\pi_1(\pi_2(\mathsf{dec}(z_0, k_{as}))), \pi_2(\mathsf{dec}(z_0, k_{as})), \mathsf{dec}(z_0, k_{as}), \pi_2(\mathsf{dec}(z_0, k_{bs})), \mathsf{dec}(z_0, k_{bs})\}$
$\mathcal{E}_1(P'_Y) = \{\mathsf{enc}(\langle z_{1\cdot 2}, \langle z_1, \mathtt{x}\rangle\rangle, k_{as}, r_s)\}$
$\overline{\mathcal{E}_1}(P'_Y) = \{\pi_2(\pi_2(\mathsf{dec}(z_0, k_{as}))), \pi_2(\mathsf{dec}(z_0, k_{as})), \mathsf{dec}(z_0, k_{as})\}$
and $\mathcal{E}_i(P'_Y) = \emptyset$ for $i \ge 2$.

Note that $\mathcal{E}(P) = \cup_{i\ge 0}\mathcal{E}_i(P)$ is finite up-to renaming of the variables since for every $i \ge 1$, every term $M \in \mathcal{E}_i(P)$, $\mathrm{pos}(M)$ is included in the (finite) set of positions occurring in terms of $\mathcal{M}_0$.

We can now define an over-approximation of the set of tests that may be applied over the secret.

$$\begin{array}{rl}\mathcal{M}_t^{\mathtt{s}}(P) = & \{T \in \mathcal{M}_t(P) \mid T = \mathtt{s} \text{ or } \exists p \in \mathrm{pos_v}(T) \text{ s.t. } D_1(\dots D_n) = f_d(T, p) \ne \perp,\\ & \exists E \in \mathcal{E}(P), \exists i \text{ s.t. } D_i = \pi^i(\mathsf{dec_g}(z_0, K)), E = \mathsf{enc_g}(U, K, R) \text{ and } \mathtt{x} \in D_i(E)\!\downarrow\}\end{array}$$

For example, $\mathcal{M}_t^{\mathtt{s}}(P'_Y) = \{\pi_1(\pi_2(\pi_2(\mathsf{dec}(\pi_1(y_a), k_{as}))))\}$.

Definition 4.23 ("no test on the secret" process) *A well-formed process P w.r.t.* $\mathtt{s}$ does not test over $\mathtt{s}$ *if the following conditions are satisfied:*

1. *for all $E \in \mathcal{E}(P)$, for all $D = D_1(\dots D_n) \in \mathcal{D}_o(P)$, if $D_i = \pi^i(\mathsf{dec_g}(z_0), K)$ and $E = \mathsf{enc_g}(U, K, R)$ and $\mathtt{x} \in \mathrm{var}(D_i(E)\!\downarrow)$ then $i = 1$ and $\overline{E} \not<_{st} D_1$,*

2. *if $[T = T']$, $[T' = T]$, $[\mathsf{check}(T, T', K) = \mathsf{ok}]$ or $[\mathsf{check}(T', T, K) = \mathsf{ok}]$ is a test of P and $T \in \mathcal{M}_t^{\mathtt{s}}(P)$ then T' is a restricted name different from* $\mathtt{s}$.

For example, P'_Y does not test over $\mathtt{s}$. Note that $\mathcal{E}(P)$ can be computed in polynomial time from P and that whether P *does not test over* $\mathtt{s}$ is decidable. We show in the next section that the first condition is sufficient to ensure that frames obtained from P are extended well-formed. It ensures in particular that there are no destructors right above $\mathtt{s}$. Indeed, informally, if some D_i cancels some encryption in some E and $\mathtt{x} \in \mathrm{var}(D_i(E)\!\downarrow)$ then all its destructors should reduce in the normal form computation (otherwise some destructors (namely projections from D_i) remain above x). Also we have $i = 1$ since otherwise a D_i may have consumed the lowest encryption above x, thus the other decryption may block, and again there would be destructors left above x.

The second condition requires that whenever an operand of a test $[T = T']$ is potentially dangerous (that is T or T' is in $\mathcal{M}_t^{\mathsf{s}}(P)$) then the other operand should be a restricted name.

Exemple 4.24 *A simple class of protocols that do not test on the secret is the one where in all messages sent by the protocol, the secret occurs only in the second component of pairs, and the tests apply only on the first component of pairs. For example, if for a protocol P_4 we have*

$$\mathcal{M}_o(P_4) = \{\mathsf{enc}(\langle n_a, \mathsf{s}\rangle, k, r), \mathsf{enc}(\langle n_a, \pi_2(\mathsf{dec}(z, k)), k', r'\rangle)\}$$

and the test is $[\pi_1(\mathsf{dec}(z', k')) = \pi_1(\mathsf{dec}(z'', k))]$ then there will be no test on s*. Moreover, this protocol also satisfies the first condition and hence we obtain that* s *is a strong secret using the main result of this section.*

We also give examples of protocols not satisfying the two conditions of Definition 4.23. Consider first a protocol P_5 for which

$$\mathcal{M}_o(P_5) = \{\mathsf{enc}(\pi_1(\mathsf{dec}(z, k)), k, r'), \mathsf{enc}(\mathsf{s}, k, r)\}.$$

P_5 does not satisfy the first condition of the previous definition because the term $\mathsf{enc}(\pi_1(\mathsf{s}), k, r)$ (with a destructor right above s*) could be obtained by sending the first message to the agent which constructs the second message.*

A second example of protocol not satisfying the conditions (this time the second one) is inspired from the Otway-Rees protocol. Consider a protocol P_6 where the server waits for $A, \{\!|N_a, A|\!\}_{K_{as}}$, performs a test on A and then sends $\{\!|N_a, K_{ab}|\!\}_{K_{as}}$. Using a second session, the intruder is able to transform the test that the server does on A into a test on the secret. Formally, the outputs are

$$\mathcal{M}_o(P_6) = \{\langle a, \mathsf{enc}(\langle n_a, a\rangle, k_{as}, r)\rangle, \mathsf{enc}(\langle \pi_1(\mathsf{dec}(\pi_2(z), k_{as})), \mathsf{s}\rangle), k_{as}, r'\}$$

and the process modeling the first actions of the server is $c(z).[\pi_1(z) = \pi_2(\mathsf{dec}(\pi_2(z), k_{as}))]$. Then $\pi_2(\mathsf{dec}(\pi_2(z), k_{as})) \in \mathcal{M}_t^{\mathsf{s}}(P_6)$, but $\pi_1(z)$ is not a restricted name.

4.3.2 Main result

We are now ready to prove that simple secrecy is actually equivalent to strong secrecy for protocols that are well-formed and do not test over the secret.

Theorem 4.25 *Let P be well-formed process w.r.t. a bound name* s *such that P does not test over* s*. We have $\varphi \nvdash \mathsf{s}$ for any valid frame φ w.r.t. P if and only if $P[{}^{M}/_{\mathsf{s}}] \approx_l P[{}^{M'}/_{\mathsf{s}}]$, for all ground terms M, M' public w.r.t. $\mathrm{bn}(P)$.*

The remaining of the section is devoted to the proof of the theorem.

Consider first the simpler implication, that is strong secrecy implies simple secrecy. Suppose that there is a valid frame φ w.r.t. P such that $\varphi \vdash \mathsf{s}$. Then, as for the passive case, there are M and M' public ground terms such that $\varphi[{}^{M}/_{\mathsf{s}}] \not\approx_s \varphi[{}^{M'}/_{\mathsf{s}}]$. Since φ is a valid frame there is an extended process A such that $P \Rightarrow^* A$ and $\varphi = \varphi(A)$. Then clearly $P[{}^{M}/_{\mathsf{s}}] \Rightarrow^* A[{}^{M}/_{\mathsf{s}}]$ and $P[{}^{M'}/_{\mathsf{s}}] \Rightarrow^* A[{}^{M'}/_{\mathsf{s}}]$. Thus if $P[{}^{M}/_{\mathsf{s}}] \approx_l P[{}^{M'}/_{\mathsf{s}}]$ then $A[{}^{M}/_{\mathsf{s}}] \approx_l A[{}^{M'}/_{\mathsf{s}}]$ and moreover $\varphi(A[{}^{M}/_{\mathsf{s}}]) \approx_s \varphi(A[{}^{M'}/_{\mathsf{s}}])$. Since $\varphi(A[{}^{T}/_{x}]) = \varphi(A)[{}^{T}/_{x}]$ for any term T, we get $\varphi[{}^{M}/_{\mathsf{s}}] \approx_s \varphi[{}^{M'}/_{\mathsf{s}}]$, contradiction. We deduce $P[{}^{M}/_{\mathsf{s}}] \not\approx_l P[{}^{M'}/_{\mathsf{s}}]$ and thus s is not a strong secret in P.

Consider now the converse implication. Let P be well-formed process w.r.t. a bound name s with no test over s and assume that s is a simple secret in P. Let M, M' be two public terms w.r.t. $\mathrm{bn}(P)$. To prove that $P[{}^{M}/_{\mathsf{s}}]$ and $P[{}^{M'}/_{\mathsf{s}}]$ are labeled bisimilar, we need to show that each move of $P[{}^{M}/_{\mathsf{s}}]$ can be matched by a move in $P[{}^{M'}/_{\mathsf{s}}]$ such that the corresponding frames are bisimilar (and conversely). By hypothesis, s is a simple secret in P thus for any valid frame φ w.r.t. P, we have $\varphi \nvdash \mathsf{s}$. In order to apply our previous result in the passive setting (Theorem 4.12), we need to show that all the valid frames are well-formed. However, frames may now contain destructors in particular if the adversary sends messages that contain destructors. That is why we consider *extended well-formed frames*, defined in Section 4.2.2.

Theorem 4.12 can easily be generalised to extended well-formed frames.

Proposition 4.26 *Let φ be an extended well-formed frame w.r.t. s, where s is a restricted name in φ. Then $\varphi \nvdash \mathsf{s}$ if and only if $\varphi[{}^{M}/_{\mathsf{s}}] \approx_s \varphi[{}^{M'}/_{\mathsf{s}}]$ for all M, M' closed public terms w.r.t. φ.*

The proof of Proposition 4.26 is exactly the same as the proof of Theorem 4.12 except that it uses Corollary 4.20 and Lemma 4.21 instead of Lemmas 4.13 and 4.14 respectively.

The first step of the proof of Theorem 4.25 is to show that any frame produced by the protocol is an extended well-formed frame. We actually prove directly a stronger result, crucial in the proof: the secret s always occurs under an agent encryption and this encryption is an instance of a term in $\mathcal{E}(P)$. This shows that $\mathcal{E}(P)$ is indeed an approximation of the ciphertexts that may contain the secret.

Lemma 4.27 *Let P be a well-formed process with no test over s and $\varphi = \nu\widetilde{n}.\sigma$ be a valid frame w.r.t. P such that $\varphi \nvdash \mathsf{s}$. Consider the corresponding standard frame $\nu\widetilde{n}.\overline{\sigma} = \nu\widetilde{n}.\{{}^{U_i}/_{y_i} \mid 1 \leq i \leq l\}$. For every i and every occurrence q_{s} of s in $U_i\!\downarrow$, we have $f_e(U_i\!\downarrow, q_{\mathsf{s}}) = E[{}^{W}/_{\mathsf{x}}]$ for some $E \in \mathcal{E}(P)$ and some term W. In addition $\nu\widetilde{n}.\sigma_i\!\downarrow$ is an extended well-formed frame w.r.t. s.*

The lemma is proved in Section 4.3.3. The proof uses an induction on i and relies deeply on the construction of $\mathcal{E}(P)$.

The second step of the proof consists in showing that any successful test in the process $P[{}^{M}/_{\mathsf{s}}]$ is also successful in P and thus in $P[{}^{M'}/_{\mathsf{s}}]$.

Lemma 4.28 *Let P be a well-formed process with no test over s, $\varphi = \nu\widetilde{n}.\sigma$ a valid frame for P such that $\varphi \nvdash \mathsf{s}$, θ a public substitution and M a public ground term. If $T_1 = T_2$ is a test in P, then $T_1\theta\sigma[{}^{M}/_{\mathsf{s}}] =_E T_2\theta\sigma[{}^{M}/_{\mathsf{s}}]$ implies $T_1\theta\sigma =_E T_2\theta\sigma$.*

This lemma is proved in Section 4.3.3 by case analysis, depending on whether $T_1, T_2 \in \mathcal{M}_t^{\mathsf{s}}(P)$ and whether s occurs or not in $\mathrm{names}(T_1\theta\sigma)$ and $\mathrm{names}(T_2\theta\sigma)$.

Using Lemmas 4.27 and 4.28, we are ready to complete the proof of Theorem 4.25, showing that $P[{}^{M}/_{\mathsf{s}}]$ and $P[{}^{M'}/_{\mathsf{s}}]$ are labeled bisimilar.

We consider the relation $\mathcal{R}$ between closed extended processes defined as follows: $A\,\mathcal{R}\,B$ if there is an extended process A_0 and ground terms M, M' public w.r.t. $\mathrm{bn}(P)$ such that $P \Rightarrow^* A_0$, $A = A_0[{}^{M}/_{\mathsf{s}}]$ and $B = A_0[{}^{M'}/_{\mathsf{s}}]$.

We show that $\mathcal{R}$ satisfies the three points of the definition of labeled bisimilarity. Suppose $A\,\mathcal{R}\,B$, that is $A_0[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,A_0[{}^{M'}/_{\mathsf{s}}]$ for some A_0, M, M' as above.

1. Let us show that $\varphi(A_0[{}^{M}/_{\mathsf{s}}]) \approx_s \varphi(A_0[{}^{M'}/_{\mathsf{s}}])$. We know that $\varphi(A_0)$ is a valid frame w.r.t. P (from the definition of $\mathcal{R}$), hence $\varphi(A_0) \nvdash \mathsf{s}$ (from the hypothesis). Let $\varphi' \equiv \varphi(A_0)$ having only ground and normalised terms (take for example $\varphi' = \overline{\varphi(A)}\!\downarrow$, where $\overline{\varphi(A)}$ is the standard frame w.r.t. A). Then, by Lemma 4.27, we have that φ' is an extended well-formed frame. We can then use Proposition 4.26 to obtain that $\varphi(A_0[{}^{M}/_{\mathsf{s}}]) \approx_s \varphi(A_0[{}^{M'}/_{\mathsf{s}}])$.

2. Let us show that if $A_0[{}^{M}/_{\mathsf{s}}] \to A'$ then $A' \equiv A'_0[{}^{M}/_{\mathsf{s}}]$, $A_0[{}^{M'}/_{\mathsf{s}}] \to A'_0[{}^{M'}/_{\mathsf{s}}]$, and $A'_0[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,A'_0[{}^{M'}/_{\mathsf{s}}]$, for some A'_0. We distinguish two cases, according to whether the transition rule was the COMM rule or one of the THEN and ELSE rules:
   - if the COMM rule was used then $A_0[{}^{M}/_{\mathsf{s}}] \equiv C[{}^{M}/_{\mathsf{s}}]\big[\overline{c}\langle z\rangle.Q[{}^{M}/_{\mathsf{s}}]|c(z).R[{}^{M}/_{\mathsf{s}}]\big]$, where C is an evaluation context and $A' = C[{}^{M}/_{\mathsf{s}}]\big[Q[{}^{M}/_{\mathsf{s}}]|R[{}^{M}/_{\mathsf{s}}]\big]$. Then $A_0 \equiv C[\overline{c}\langle z\rangle.Q|c(z).R]$. Take $A'_0 = C[Q|R]$. We have that $P \Rightarrow^* A'_0$ and thus, by definition of $\mathcal{R}$, we have that $A'_0[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,A'_0[{}^{M'}/_{\mathsf{s}}]$.
   - otherwise, $A_0[{}^{M}/_{\mathsf{s}}] \equiv C[{}^{M}/_{\mathsf{s}}]\big[\text{if } T'[{}^{M}/_{\mathsf{s}}] = T''[{}^{M}/_{\mathsf{s}}] \text{ then } Q[{}^{M}/_{\mathsf{s}}] \text{ else } R[{}^{M}/_{\mathsf{s}}]\big]$. Then $A_0 \equiv C[\text{if } T' = T'' \text{ then } Q \text{ else } R]$. From Lemma 4.4 we know that $T' = T'_0\theta\sigma$ and $T'' = T''_0\theta\sigma$, where $T'_0 = T''_0$ is a test in P and $\nu\widetilde{n}.\sigma \equiv \varphi(A_0)$ is the standard frame w.r.t. A_0. Take $A'_0 = C[Q]$ if $T'_0\theta\sigma =_E T''_0\theta\sigma$ and $A'_0 = C[R]$ otherwise. From Lemma 4.28 we have that $T'_0\theta\sigma =_E T''_0\theta\sigma$ if and only if $T'_0\theta\sigma[{}^{M}/_{\mathsf{s}}] =_E T''_0\theta\sigma[{}^{M}/_{\mathsf{s}}]$. Hence $A_0[{}^{M}/_{\mathsf{s}}] \to A'_0[{}^{M}/_{\mathsf{s}}]$, $A_0[{}^{M'}/_{\mathsf{s}}] \to A'_0[{}^{M'}/_{\mathsf{s}}]$ and $A_0 \to A'_0$. We conclude $A'_0[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,A'_0[{}^{M'}/_{\mathsf{s}}]$ from the definition of $\mathcal{R}$.
3. Let us show that if $A_0[{}^{M}/_{\mathsf{s}}] \xrightarrow{\alpha} A'$ and $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(\varphi(A_0[{}^{M}/_{\mathsf{s}}]))$ and $\mathrm{bn}(\alpha)\cap\mathrm{fn}(A_0[{}^{M'}/_{\mathsf{s}}]) = \emptyset$ then $A' \equiv A'_0[{}^{M}/_{\mathsf{s}}]$, $A_0[{}^{M'}/_{\mathsf{s}}] \xrightarrow{\alpha} A'_0[{}^{M'}/_{\mathsf{s}}]$ and $A'_0[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,A'_0[{}^{M'}/_{\mathsf{s}}]$, for some A'_0. Depending on the form of α, we consider the following cases:
   - $\alpha = c(T)$. Suppose $A_0[{}^{M}/_{\mathsf{s}}] \equiv C[{}^{M}/_{\mathsf{s}}]\big[c(z).Q[{}^{M}/_{\mathsf{s}}]\big]$. Then take $A'_0 = C[Q\{{}^{T}/_{z}\}]$.
   - $\alpha = \overline{c}\langle u\rangle$. Suppose $A_0[{}^{M}/_{\mathsf{s}}] \equiv C[{}^{M}/_{\mathsf{s}}]\big[\overline{c}\langle u\rangle.Q[{}^{M}/_{\mathsf{s}}]\big]$. Then take $A'_0 = C[Q]$.
   - $\alpha = \nu u.\overline{c}\langle u\rangle$. Suppose $A_0[{}^{M}/_{\mathsf{s}}] \equiv C[{}^{M}/_{\mathsf{s}}]\big[\nu u.A_1[{}^{M}/_{\mathsf{s}}]\big]$, where $A_1[{}^{M}/_{\mathsf{s}}] \xrightarrow{\overline{c}\langle u\rangle} A'_1[{}^{M}/_{\mathsf{s}}]$. Then take $A'_0 = C[A_1]$.

The above discussion proves that $\mathcal{R} \subseteq \approx_l$. Since we have $P[{}^{M}/_{\mathsf{s}}]\,\mathcal{R}\,P[{}^{M'}/_{\mathsf{s}}]$ it follows that $P[{}^{M}/_{\mathsf{s}}] \approx_l P[{}^{M'}/_{\mathsf{s}}]$.

4.3.3 Proofs of intermediate results

In what follows we usually simply write $\mathcal{M}$, $\mathcal{M}_t$, $\mathcal{M}_o$, $\mathcal{D}_o$, $\mathcal{E}$ instead of respectively $\mathcal{M}(P)$, $\mathcal{M}_t(P)$, $\mathcal{M}_o(P)$, $\mathcal{D}_o(P)$, $\mathcal{E}(P)$, *etc.*

We also define the partial subtraction function $- : \mathbb{N}_+^* \times \mathbb{N}_+^* \to \mathbb{N}_+^*$ as follows: $p - q = r$ if $p = q \cdot r$ and $p - q = \perp$ otherwise.

Let U and V be two terms. We define $\mathrm{pos}(U, V) = \{p \in \mathrm{pos}(U) \mid U|_p = V\}$.

Observe that for the rewriting system corresponding to equational theory E, there is at most one rule that can be applied and for each rule $L \to R$, there is exactly one occurrence of R in L.

We denote by $U \to^q V$ the reduction $U \to V$ such that $U|_q = L\theta$ and $V = U[R\theta]_q$, where q is a position in U, $L \to R$ is a rule in $\mathcal{R}(E)$, and θ is a substitution. Let p be a position in U. We define a partial function $\mathrm{par}_1(U, p, q)$ that computes, when $U \to^q V$, the *position after one rewriting* of a function symbol at position p in U. In particular, if $\mathrm{par}_1(U, p, q) \neq \perp$ then $U|_p = V|_{\mathrm{par}_1(U,p,q)}$. Formally, we define the function $\mathrm{par}_1 : \mathcal{T} \times \mathbb{N}_+^* \times \mathbb{N}_+^* \to \mathbb{N}_+^*$ as follows:

$$\mathrm{par}_1(U, p, q) = \begin{cases} p', & \text{if } U \to^q V \\ \perp, & \text{otherwise,} \end{cases}$$

where

$$p' = \begin{cases} p, & \text{if } p \not\geq q, \\ \perp, & \text{if } p \geq q \ \wedge\ p \not\geq q \cdot q_r, \\ q \cdot (p - q \cdot q_r), & \text{if } p \geq q \cdot q_r, \end{cases}$$

and $L \to R$ is the rule that was applied and q_r is the position of R in L.

Similarly, the function $\mathrm{par}(U, p)$ computes the *position after rewriting* in $U\!\downarrow$. The function $\mathrm{par}\colon \mathcal{T} \times \mathbb{N}_+^* \hookrightarrow \mathbb{N}_+^*$ is formally defined by $\mathrm{par}(U, p) = p_k$ where $U \to^{q_1} \cdots \to^{q_k} U_k$, $U_k = U\!\downarrow$, $p_i = \mathrm{par}_1(U, p_{i-1}, q_i)$, for $1 \le i \le k$ and $p_0 = p$. Due to the particular form of our equational theory, the choice of the rewriting steps does not change the final value of p_k thus the definition is correct.

The function $\mathrm{par}^{-1}(U, p)$ is the inverse function: to a position p in $U\!\downarrow$ it associates the corresponding position in U, that is, $\mathrm{par}^{-1}\colon \mathcal{T} \times \mathbb{N}_+^* \hookrightarrow \mathbb{N}_+^*$, $\mathrm{par}^{-1}(U, p) = p'$ if and only if $\mathrm{par}(U, p') = p$.

We say that a function symbol at position p is *consumed in V w.r.t. the reduction $U \to^q V$* if $\mathrm{par}_1(U, p, q)$ is undefined. Similarly, we say that a function symbol at position p is *consumed in $U\!\downarrow$ w.r.t. the normal form $U\!\downarrow$* if $\mathrm{par}(U, p)$ is undefined. We say simply that an occurrence is consumed in some term when it is clear from the context which definition is used.

Lemma 4.27 *Let P be a well-formed process with no test over s and $\varphi = \nu\widetilde{n}.\sigma$ be a valid frame w.r.t. P such that $\varphi \nvdash \mathsf{s}$. Consider the corresponding standard frame $\nu\widetilde{n}.\overline{\sigma} = \nu\widetilde{n}.\{{}^{U_i}/_{y_i} \mid 1 \le i \le l\}$. For every i and every occurrence q_{s} of s in $U_i\!\downarrow$, we have $f_e(U_i\!\downarrow, q_{\mathsf{s}}) = E[{}^{W}/_{\mathtt{x}}]$ for some $E \in \mathcal{E}(P)$ and some term W. In addition $\nu\widetilde{n}.\sigma_i\!\downarrow$ is an extended well-formed frame w.r.t. s.*

Proof We write the standard frame $\overline{\sigma}$ as in the statement of Lemma 4.4, that is $U_i = M_i\theta_i\sigma_{i-1}$ for all $1 \le i \le l$ with M_i an output in P, θ_i a public substitution w.r.t s and $\sigma_i = \sigma_{i-1} \uplus \{{}^{U_i}/_{y_i}\}$, σ_0 being the empty substitution. We reason by induction on i.

Base case: $i = 1$. We have that $U_1 = M_1\theta_1$. Then $U_1\!\downarrow = M_1(\theta_1\!\downarrow)$ since there are no destructors in the output M_1. Hence any position q_{s} of s is in fact a position in M_1 since s cannot appear in θ_1 because s is restricted and θ is a public substitution. There must an encryption above q_{s} in M_1 (that is a position $q_{\mathsf{enc}} \cdot 1 \le q_{\mathsf{s}}$), since otherwise s would be deducible (the same argument as in Lemma 4.17 applies). Then the result follows immediately from the definition of $\mathcal{E}_0$ (take $W = \mathsf{s}$) and the properties of well-formed processes.

Inductive step. Let $p_{\mathsf{s}} = \mathrm{par}^{-1}(U_i, q_{\mathsf{s}})$.

If $p_{\mathsf{s}} \in \mathrm{pos}(M_i)$ then, as in the previous paragraph, $f_e(U_i\!\downarrow, q_{\mathsf{s}})[{}^{\mathtt{x}}/_{\mathsf{s}}] \in \mathcal{E}_0$.

Otherwise, since θ_i is public, $p_{\mathsf{s}} \notin \mathrm{pos}(M_i\theta)$. It follows that there are $z \in \mathrm{var}(M_i)$ and $y_{i_1} \in \mathrm{var}(M_i\theta_i)$ at positions p_z and p_{y_1} respectively, such that $p_z \le p_{y_1} \le p_{\mathsf{s}}$ and $1 \le i_1 \le i-1$. Let $p_{\mathsf{s}}^1 = p_{\mathsf{s}} - p_{y_1}$ and $q_{\mathsf{s}}^1 = \mathrm{par}(U_{i_1}, p_{\mathsf{s}}^1)$. By induction hypothesis, σ_{i-1} is an extended well-formed frame and $f_e(U_{i_1}\!\downarrow, q_{\mathsf{s}}^1) = E[{}^{W}/_{\mathtt{x}}]$ with $E \in \mathcal{E}_l$, for some term W and some $l \ge 0$. It follows from the definition of extended well-formed frames that in $y_1\sigma_{i_1}$ there is an encryption above q_{s}^1, that is $q_{\mathsf{enc}}^1 = \max\{\, q \in \mathrm{pos}(U_{i_1}\!\downarrow) \mid q < q_{\mathsf{s}}^1 \ \wedge\ \mathrm{head}((U_{i_1}\!\downarrow)|_q) = \mathsf{enc_g} \,\}$ exists. Let $p_{\mathsf{enc}}^1 = \mathrm{par}^{-1}(U_{i_1}, q_{\mathsf{enc}}^1)$.

If $p_{y_1} \cdot p_{\mathsf{enc}}^1$ is not consumed in $U_i\!\downarrow$ then $\mathrm{par}(U_i, p_{y_1} \cdot p_{\mathsf{enc}}^1)$ is the lowest encryption in $U_i\!\downarrow$ above q_{s}^1 (since it corresponds to q_{enc}^1). It follows that $f_e(U_i\!\downarrow, q_{\mathsf{s}}) = f_e(U_{i_1}\!\downarrow, q_{\mathsf{s}}^1)$.

Otherwise, that is if $p_{y_1} \cdot p_{\mathsf{enc}}^1$ is consumed in $U_i\!\downarrow$, consider the occurrence of $\mathsf{dec_g}$ in U_i, say p_{dec}, that consumes it. Since p_{enc}^1 is not consumed w.r.t. $U_{i_1}\!\downarrow$ it follows that $p_{\mathsf{dec}} \in \mathrm{pos}(M_i\theta_i)$, and all encryptions above p_{enc}^1 in U_{i_1} are consumed in $U_i\!\downarrow$. If p_{dec} is in $z\theta_i$ (that is, $p_{\mathsf{dec}} \notin \mathrm{pos_{nv}}(M_i)$) then all encryptions above p_{enc}^1 in U_{i_1} are consumed by decryptions that are in $z\theta_i$. This means that in $(z\theta_i\sigma_{i-1})\!\downarrow$ there is no encryption above s and thus $\varphi \vdash \mathsf{s}$. Hence p_{dec} is in M_i (that is, $p_{\mathsf{dec}} \in \mathrm{pos_{nv}}(M_i)$).

Let U, V, K, K' and R be terms such that $\mathsf{dec_g}(U, K) = U_i|_{p_{\mathsf{dec}}}$ and $\mathsf{enc_g}(V, K', R) = U_i|_{p_{y_1} \cdot p_{\mathsf{enc}}^1} = U_{i_1}|_{p_{\mathsf{enc}}^1}$. We have that $K =_E K'$ since p_{dec} consumes $p_{y_1} \cdot p_{\mathsf{enc}}^1$. We then have $\mathsf{dec_g}(U, K) \to^* \mathsf{dec_g}(\mathsf{enc_g}(V, K, R), K) \to^* V\!\downarrow$.

Let $(D, p) = f_{dp}(M_i, p_z)$ and write it as $D = D_1(\dots D_n)$ where $D_j = \pi^j(\mathsf{dec_g}(\mathrm{z}_0, K_j))$ with $1 \leq j \leq n$ and consider D_k such that the decryption p_{dec} is that of D_k. Clearly $\mathtt{x} \in \mathrm{var}(D_j(E)\downarrow)$. From the first condition of processes that do not test over s we have that $j = 1$ and $\overline{E} \not<_{st} D_1$. Since p_{dec} consumes $p_{y_1} \cdot p^1_{\mathsf{enc}}$, above p_{dec} in D_1 there are only projections, below $\mathsf{enc_g}$ in E there are only pairs and $\overline{E} \not<_{st} D_1$ it follows that $D_1 \leq_{st} \overline{E}$. Hence $D_1 \in \overline{\mathcal{E}}_l$.

Suppose that there is no encryption above p_{dec} in M_i. Then since D_1 is consumed and above D_1 in M_i there are only pairs or signatures, it follows that s is deducible from σ_i (more exactly from $U_i\downarrow$). Thus there is at least one encryption above p_{dec} in M_i. Let $(M', p_{\mathsf{enc}}) = f_{ep}(M_i, p_z)$. Then $M'[\mathtt{x}]_p \in \mathcal{E}_{l+1}$.

Since p_{enc} is not consumed in $U_i\downarrow$, and in M' all function symbols above p are not destructors we have that $f_e(U_i, p_{\mathsf{s}}) \rightarrow^* (M'[\mathtt{x}]_p)[^{W'}/_{\mathtt{x}}]$ with $p'_{\mathsf{s}} = p^1_{\mathsf{s}} - p^1_{\mathsf{enc}}$ and $W' = D_1(f_e(\mathsf{enc_g}(V, K', R), p'_{\mathsf{s}}))\downarrow$. Hence $f_e(U_i\downarrow, q_{\mathsf{s}}) = (M'[\mathtt{x}]_p)[^{W'}/_{\mathtt{x}}]$. That is we have the first part of the lemma.

In order to prove that $\sigma\downarrow$ is an extended well-formed frame we just need to show that $M'[\mathtt{x}]_p$ and W' contain only pairs and signatures (except for the head of $M'[\mathtt{x}]_p$ which is an encryption); obviously all agent encryptions are probabilistic encryption, either by the definition of well-formed process or by induction hypothesis. From the definition of M' all function symbols (except for the head) in $M'[\mathtt{x}]_p$ are pairs and signatures. And since σ_{i_1} is an extended well-formed frame and the term W' is a subterm of $f_e(\mathsf{enc_g}(V\downarrow, K', R), q'_{\mathsf{s}})$ which (except for the head) contains only pairs as function symbols and signatures by definition of f_e. ■

Lemma 4.29 *Let P be a well-formed process with no test over* s*, $\varphi = \nu\widetilde{n}.\sigma$ be a valid frame w.r.t. P such that $\varphi \nvdash \mathsf{s}$, $T \in \mathcal{M}_t(P)$ be an operand of a test and θ be a public substitution. If $T \notin \mathcal{M}^{\mathsf{s}}_t$ then for any occurrence q_{s} of* s *in $(T\theta\sigma)\downarrow$ there is an encryption q_{enc} plaintext-above it such that this encryption is an agent encryption w.r.t. $\widetilde{n}\backslash\{\mathsf{s}\}$, is a probabilistic encryption w.r.t.* $\mathrm{ran}(\sigma)$ *and* $\mathrm{head}((T\theta\sigma)\downarrow|_q) \in \{\langle\rangle, \mathsf{sign}\}$*, for all positions q with $q_{\mathsf{enc}} < q < q_{\mathsf{s}}$.*

Proof Suppose that $T \notin \mathcal{M}^{\mathsf{s}}_t$ and consider an occurrence q_{s} of s in $(T\theta\sigma)\downarrow$. Hence T is not ground and denote by z the variable of T and by p_z its position. Let $T_z = (z\theta\sigma)\downarrow$.

Let $\overline{\sigma} = \{^{U_1}/_{y_1}, \dots, ^{U_l}/_{y_l}\}$ be the standard frame w.r.t. A (where $\varphi = \varphi(A)$ for some extended process A). Let $p_{\mathsf{s}} = \mathrm{par}^{-1}(T\theta\overline{\sigma}, q_{\mathsf{s}})$. Let y_i be the variable of $z\theta$ on the path to p_{s} at position say p_y, with $1 \leq i \leq l$. Applying Lemma 4.27 to U_i we obtain that $f_e(U_i\downarrow, q_{\mathsf{s}}) = E[^W/_{\mathtt{x}}]$ with $E \in \mathcal{E}(P)$, for some term W. Consider the lowest encryption q_{enc} in $U_i\downarrow$ above q'_{s}, where q'_{s} is the position in $U_i\downarrow$ of q_{s}.

Suppose that this encryption is consumed. Then it must be consumed by a $\mathsf{dec_g}$ from T since otherwise s would be deducible. It follows that there is $1 \leq j \leq l$ such that $D_j = \pi^j(\mathsf{dec}(\mathrm{z}_0, K))$, where $f_d(T, p_z) = D_1(\dots D_n)$, $E = \mathsf{enc}(U, K, R)$ and $\mathtt{x} \in D_i(E)\downarrow$ for some terms U, K and R. Thus $T \in \mathcal{M}^{\mathsf{s}}_t$, but this contradicts the hypothesis. Hence q_{enc} is not consumed in $(T\theta\sigma)\downarrow$. Since $\nu\widetilde{n}.\sigma\downarrow$ is an extended well-formed frame (again from Lemma 4.27) then the encryption q_{enc} clearly satisfies the hypothesis. ■

Lemma 4.28 *Let P be a well-formed process with no test over* s*, $\varphi = \nu\widetilde{n}.\sigma$ a valid frame for P such that $\varphi \nvdash \mathsf{s}$, θ a public substitution and M a public ground term. If $T_1 = T_2$ is a test in P, then $T_1\theta\sigma[^M/_{\mathsf{s}}] =_E T_2\theta\sigma[^M/_{\mathsf{s}}]$ implies $T_1\theta\sigma =_E T_2\theta\sigma$.*

Proof $T_1\theta\sigma[^M/_{\mathsf{s}}] =_E T_2\theta\sigma[^M/_{\mathsf{s}}]$ rewrites in $(T_1\theta\sigma[^M/_{\mathsf{s}}])\downarrow = (T_2\theta\sigma[^M/_{\mathsf{s}}])\downarrow$. Since the rewrite system $\mathcal{R}(E)$ is convergent, it follows that $((T_1\theta\sigma)\downarrow[^M/_{\mathsf{s}}])\downarrow = ((T_2\theta\sigma)\downarrow[^M/_{\mathsf{s}}])\downarrow$.

Suppose first that $T_1, T_2 \notin \mathcal{M}_t^{\mathsf{s}}$. Then from Lemma 4.29 right above any occurrence of s in $(T_1\theta\sigma){\downarrow}$ there are no destructors, hence $(T_1\theta\sigma){\downarrow}[{}^{M}/_{\mathsf{s}}]$ is already in normal form. The same thing holds for T_2. Thus $(T_1\theta\sigma){\downarrow}[{}^{M}/_{\mathsf{s}}] = (T_2\theta\sigma){\downarrow}[{}^{M}/_{\mathsf{s}}]$. Lemma 4.29 also ensures that in $(T_1\theta\sigma){\downarrow}$ and $(T_2\theta\sigma){\downarrow}$ there is an agent probabilistic encryption above each occurrence of s. Hence we can apply Lemma 4.19 and obtain that $(T_1\theta\sigma){\downarrow} = (T_2\theta\sigma){\downarrow}$, that is $T_1\theta\sigma =_E T_2\theta\sigma$.

Suppose now that $T_1 \in \mathcal{M}_t^{\mathsf{s}}$. Then $T_2 = n$ where n is a restricted name. The name n is a subterm of $(T_1\theta\sigma[{}^{M}/_{s}]){\downarrow}$ appearing at a position p in $T_1\theta\sigma[{}^{M}/_{s}]$. Since M is public, while T_2 is restricted it follows n is not a subterm of M, that is there is no occurrence q_{s} of s in $T_1\theta\sigma$ such that $q_{\mathsf{s}} \leq p$. Then $((T_1\theta\sigma){\downarrow}[{}^{M}/_{s}]){\downarrow} = (T_1\theta\sigma){\downarrow}[{}^{M}/_{s}]$. Hence $(T_1\theta\sigma){\downarrow} = n$.

If the test is $\mathsf{check}(T, T', K) = \mathsf{ok}$ then $T\theta\sigma[{}^{M}/_{\mathsf{s}}] =_E \mathsf{retrieve}(T')\theta\sigma[{}^{M}/_{\mathsf{s}}]$. Applying the lemma for the test $T =_E \mathsf{retrieve}(T')$ we obtain that $T\theta\sigma =_E \mathsf{retrieve}(T')\theta\sigma$. Since the keys are ground then it follows that $\mathsf{check}(T, T', K)\theta\sigma =_E \mathsf{ok}$.

■

4.4 Application to some cryptographic protocols

We apply our result to three protocols (Yahalom, Needham-Schroeder with symmetric keys and Wide-Mouthed-Frog), known to preserve the usual simple secrecy property. Since all these three protocols satisfy our hypotheses, we directly deduce that they preserve the strong secrecy property.

4.4.1 Yahalom

We have seen in Section 4.3.1 that P_Y is a well-formed process w.r.t. k_{ab} and does not test over k_{ab}. Applying Theorem 4.25, if P'_Y preserves the simple secrecy of k_{ab}, we can deduce that the Yahalom protocol preserves the strong secrecy of k_{ab} that is

$$P'_Y[{}^{M}/_{k_{ab}}] \approx_l P'_Y[{}^{M'}/_{k_{ab}}]$$

for any public terms M, M' w.r.t. $\mathrm{bn}(P'_Y)$. We did not formally prove that the Yahalom protocol preserves the simple secrecy of k_{ab} but this was done with several tools in slightly different settings (e.g. [BLP03, Pau01]).

4.4.2 Needham-Schroeder symmetric key protocol

A simplified version of the Needham-Schroeder symmetric key protocol [NS78] is described below:

$$\begin{array}{rl}
A \Rightarrow S: & A, B, N_a \\
S \Rightarrow A: & \{\!\{N_a, B, K_{ab}, \{\!\{K_{ab}, A\}\!\}_{K_{bs}}\}\!\}_{K_{as}} \\
A \Rightarrow B: & \{\!\{K_{ab}, A\}\!\}_{K_{bs}}
\end{array}$$

The target secret is K_{ab}. The protocol is modeled by the following process:

$$P_{\mathsf{NS}} = \nu k_{as}.\nu k_{bs}.\,(!A) \mid (!c(y_b)) \mid (!\nu k.S(k)) \mid \nu k_{ab}.S(k_{ab})$$

where

$$\begin{array}{rcl}
A & = & \nu n_a.\overline{c}\langle a, b, n_a\rangle.c(y_a).[\pi_1(\mathsf{dec}(y_a, k_{as})) = n_a]. \\
 & & [\pi_1(\pi_2(\mathsf{dec}(y_a, k_{as}))) = b].\overline{c}\langle \pi_2(\pi_2(\pi_2(\mathsf{dec}(y_a, k_{as}))))\rangle \\
S(x) & = & c(y_s).\nu r, r'.\overline{c}\langle \mathsf{enc}(\langle \pi_2(\pi_2(y_s)), \pi_1(\pi_2(y_s)), x, \\
 & & \mathsf{enc}(\langle x, \pi_1(y_s)\rangle, k_{bs}, r')\rangle, k_{as}, r)\rangle
\end{array}$$

Note that other processes should be added to considered corrupted agents or roles A, B and S talking to other agents but this would not really change the following sets of messages.

The output messages are:

$$\mathcal{M}_o = \left\{ \begin{array}{l} a, b, n_a \\ \pi_2(\pi_2(\pi_2(\mathsf{dec}(y_a, k_{as})))) \\ \mathsf{enc}(\langle \pi_2(\pi_2(y_s)), \pi_1(\pi_2(y_s)), \\ k_{ab}, \mathsf{enc}(\langle k_{ab}, \pi_1(y_s)\rangle, k_{bs}, r')\rangle, k_{as}, r) \end{array} \right\}$$

The tests are:

$$\left\{ \begin{array}{l} \pi_1(\mathsf{dec}(y_a, k_{as})) = n_a \\ \pi_1(\pi_2(\mathsf{dec}(y_a, k_{as}))) = b \end{array} \right\}$$

We define $\max \overline{\mathcal{E}_i} = \{\overline{e} \mid e \in \mathcal{E}_i\}$ in order to increase readability, and since it is easy to deduce $\overline{\mathcal{E}_i}$ from $\max \overline{\mathcal{E}_i}$.

$$\begin{aligned} \mathcal{D}_o &= \{\pi_2(\pi_2(\pi_2(\mathsf{dec}(z, k_{as}))))\} \\ \mathcal{E}_0 &= \{\mathsf{enc}(\langle z_1, \langle z_2, \langle \mathtt{x}, z_3\rangle\rangle\rangle, k_{as}, r), \mathsf{enc}(\langle \mathtt{x}, z_4\rangle, k_{bs}, r')\} \\ \max \overline{\mathcal{E}}_0 &= \{\pi_1(\pi_2(\pi_2(\mathsf{dec}(z, k_{as})))), \pi_1(\mathsf{dec}(z, k_{bs}))\} \\ \mathcal{D}_o \cap \overline{\mathcal{E}}_0 &= \emptyset \\ \mathcal{M}_t^{k_{ab}} &= \emptyset \end{aligned}$$

We deduce that P_{NS} is a well-formed process w.r.t. k_{ab}, that does not test over k_{ab}. Applying Theorem 4.25 and since the Needham-Schroeder symmetric key protocol is known to preserve simple secrecy of k_{ab}, we deduce that the protocol preserves strong secrecy of k_{ab} that is

$$P_{\mathsf{NS}}[^{M}/_{k_{ab}}] \approx_l P_{\mathsf{NS}}[^{M'}/_{k_{ab}}]$$

for any public terms M, M' w.r.t. $\mathrm{bn}(P_{\mathsf{NS}})$.

4.4.3 Wide Mouthed Frog Protocol (modified)

We consider a modified version of the Wide Mouthed Frog Protocol [BAN90], where timestamps are replaced by nonces.

$$\begin{array}{rl} A \Rightarrow B: & N_a \\ B \Rightarrow S: & \{\!\{N_a, A, K_{ab}\}\!\}_{K_{bs}} \\ S \Rightarrow A: & \{\!\{N_a, B, K_{ab}\}\!\}_{K_{as}} \end{array}$$

The target secret is K_{ab}. The protocol is modeled by the following process:

$$P_{\mathsf{WMF}} = \nu k_{as}.\nu k_{bs}.\,(!A) \mid (!S) \mid (!\nu k.B(k)) \mid \nu k_{ab}.B(k_{ab})$$

where

$$\begin{array}{rcl} A & = & \nu n_a.\overline{c}\langle n_a\rangle.c(y_a).[\pi_1(\mathsf{dec}(y_a, k_{as})) = n_a] \\ B(x) & = & c(y_b).\nu r.\overline{c}\langle \mathsf{enc}(\langle y_b, a, x\rangle, k_{bs}, r)\rangle \\ S & = & c(y_s).[\pi_1(\pi_2(\mathsf{dec}(y_s, k_{bs}))) = a]. \\ & & \qquad \nu r'.\overline{c}\langle \mathsf{enc}(\langle \pi_1(\mathsf{dec}(y_s, k_{bs})), b, \pi_2(\pi_2(\mathsf{dec}(y_s, k_{bs})))\rangle, k_{as}, r')\rangle \end{array}$$

Note that other processes should be added to considered corrupted agents or roles A, B and S talking to other agents but again, this would not really change the following sets of messages.

The output messages are:

$$\mathcal{M}_o = \left\{ \begin{array}{l} n_a \\ \mathsf{enc}(\langle y_b, a, k_{ab}\rangle, k_{bs}, r) \\ \mathsf{enc}(\langle \pi_1(\mathsf{dec}(y_s, k_{bs})), b, \\ \pi_2(\pi_2(\mathsf{dec}(y_s, k_{bs})))\rangle, k_{as}, r') \end{array} \right\}$$

The tests are:

$$\left\{ \begin{array}{l} \pi_1(\mathsf{dec}(y_a, k_{as})) = n_a \\ \pi_1(\pi_2(\mathsf{dec}(y_s, k_{bs}))) = a \end{array} \right\}$$

$$\begin{aligned} \mathcal{D}_o &= \{\pi_1(\mathsf{dec}(z, k_{bs})), \pi_2(\pi_2(\mathsf{dec}(z, k_{bs})))\} \\ \mathcal{E}_0 &= \{\mathsf{enc}(\langle z_1, \langle z_2, \mathtt{x}\rangle, k_{bs}, r)\rangle\} \\ \max \overline{\mathcal{E}}_0 &= \{\pi_2(\pi_2(\mathsf{dec}(z, k_{bs})))\} \\ \mathcal{E}_1 &= \{\mathsf{enc}(\langle z_1, \langle z_2, \mathtt{x}\rangle, k_{as}, r)\rangle\} \\ \max \overline{\mathcal{E}}_1 &= \{\pi_2(\pi_2(\mathsf{dec}(z, k_{as})))\} \\ \mathcal{D}_o \cap \overline{\mathcal{E}}_1 &= \emptyset \\ \mathcal{M}_t^{k_{ab}} &= \emptyset \end{aligned}$$

We obtain similarly that P_{WMF} is a well-formed process w.r.t. k_{ab}, that does not test over k_{ab}. Applying Theorem 4.25 and since the Wide Mouthed Frog protocol is known to preserve simple secrecy of k_{ab}, we deduce that the protocol preserves strong secrecy of k_{ab} that is

$$P_{\mathsf{WMF}}[{}^{M}/_{k_{ab}}] \approx_l P_{\mathsf{WMF}}[{}^{M'}/_{k_{ab}}]$$

for any public terms M, M' w.r.t. $\mathrm{bn}(P_{\mathsf{WMF}})$.

4.5 Conclusions

In recent years many automatic tools have been developed for verifying security protocols. The overwhelming majority of them address reachability-based properties such as simple secrecy. On the other hand some important security notions such as strong secrecy rely on provable equivalences between systems. Typically the impossibility of guessing a vote or a password is commonly expressed that way. Hence in order to widen the scope of the current protocol analysis tools, in the present chapter we have shown how simple secrecy actually implies strong secrecy in both passive and active setting under some conditions, motivated by counterexamples. In particular such a result cannot hold for deterministic encryption and we had to assume that it is *probabilistic*.

As future work, we would like to relax our syntactic conditions. One such condition requires in the passive case that the secret does not appear in keys, and in the active case that keys are ground. This deters us from analysing for example protocols against guessing attacks. We plan to investigate whether such conditions can be replaced by more semantics ones, like asking, in the passive case, that the plaintexts encrypted by the secret are simple secrets.

We also plan to investigate whether the procedure [Bau05] for deciding static equivalence of M. Baudet can be combined with our condition on the tests in order to obtain a (non-complete) procedure allowing us to verify strong secrecy for a wider class of protocols. We would have thus to answer the challenging question of whether the exists a finite number of frames characterising (in some way) all executions of a protocol.

Chapter 5

A transformation for obtaining secure protocols

Contents

5.1 Comparison with Katz and Yung's compiler **141**

5.2 The model **142**

5.3 Security properties **144**

5.3.1 A logic for security properties 144

5.3.2 Examples of security properties 146

5.4 Transformation of protocols **147**

5.5 Transfer result **148**

5.5.1 Honest, single session traces 148

5.5.2 Transferable security properties 149

5.5.3 Transference theorem 149

5.5.4 Honest executions 150

5.5.5 Proof sketch of the transference theorem 151

5.5.6 Detailed proofs 152

5.6 Conclusions **157**

In this chapter we introduce a transformation that takes as input a protocol that is secure (in a sense that we discuss below) in a *single* execution of the protocol, with *no adversary* present (not even a passive eavesdropper). The output of the transformation is a protocol that withstands a realistic adversary with absolute control of the communication between an unbounded number of protocol sessions.

At a high level, the transformation works by binding messages to sessions using digital signatures on the concatenation of these messages with dynamically generated session identifiers, and hiding messages from the adversary using public key encryption. More specifically, the transformation is as follows. Consider a protocol with k participants $A_1, \ldots, A_k$ and n exchanges of messages.

$$\begin{array}{ll} A_{i_1} \to A_{j_1}: & m_1 \\ \vdots & \\ A_{i_n} \to A_{j_n}: & m_n \end{array}$$

The transformed protocol starts with a preliminary phase, where each participant A_i broadcasts a fresh nonce N_i to all other participants. The concatenation of the nonces with the identities of the participants forms a session identifier $\mathsf{sessionID} = \langle A_1, A_2, \ldots, A_k, N_1, N_2, \ldots, N_k \rangle$. Note that the adversary may of course interact in this preliminary phase and may send faked nonces for example. Such a behaviour would however be detected in the next phase. The remainder of the protocol works roughly as the original one except that each message is sent together with a signature on the message concatenated with the session identifier, and the whole construct is encrypted under the recipient's public key:

$$\begin{array}{ll} A_{i_1} \rightarrow A_{j_1}: & \{\![m_1, [\![m_1, p_1, \mathsf{sessionID}]\!]_{\mathsf{sk}(A_{i_1})}]\!\}_{\mathsf{ek}(A_{j_1})} \\ \quad\vdots & \\ A_{i_n} \rightarrow A_{j_n}: & \{\![m_n, [\![m_n, p_n, \mathsf{sessionID}]\!]_{\mathsf{sk}(A_{i_n})}]\!\}_{\mathsf{ek}(A_{j_n})} \end{array}$$

where the p_i's are the current control points in the participant's programs.

Intuitively, our transformation ensures that the messages of the protocol sent between honest parties in any given session of the original protocol, cannot be learned and/or blindly replayed by the adversary to unsuspecting users in other protocol sessions. Indeed, the adversary cannot impersonate users in honest sessions (since in this case it would need to produce digital signatures on their behalf), and cannot learn secrets by replaying messages from one session to another (since messages are encrypted, and any blindly replayed message would be rejected due to un-matching session identifiers).

Although the transformation does not preserve all imaginable security properties (for example, any anonymity that the original protocol might enjoy is lost due to the use of public key encryption) it does preserve several interesting properties. In particular, we exhibit a class of logic formulas which, if satisfied in single executions of the original protocol are also satisfied by the transformed protocol in the presence of active adversaries. The class that we consider includes standard formulations for secrecy and authentication (for example injective agreement [Low97] and several other variants).

Our transformation enables more modular and manageable protocol development. One can start by building a protocol with the desirable properties built-in, and bearing in mind that no adversary is actually present. Then, the final protocol is obtained using the transformation that we propose. We remark that designers can easily deal with the case of single session and it is usually the more involved setting (multi-party, many-session) that causes the real problems. Indeed, for the class of properties that we consider security verification is trivial for single, honest executions. As an example, we show how to derive a simple protocol for authentication later in the chapter.

Related work Our work is inspired by a recent compiler introduced by Katz and Yung [KY03] which transforms any group key exchange protocol secure against a passive adversary into one secure against an active adversary. Their transformation is, in some sense, simpler since they do not require that the messages in the transformed protocol are encrypted. However, their transformation is also weaker since although it requires that the protocol be secure against passive adversaries, these adversaries still can corrupt parties adaptively (even after the execution has finished). Furthermore, while their transformation is sufficient for the case of group key exchange, it fails to guarantee the transfer of more general security properties. The reason for the failure is that an adversary can obtain a message (e.g. a ciphertext) from a session with only honest participants, and get information about the message (e.g. the underlying plaintext) by replaying it in some other sessions for which he can produce the necessary digital signatures. We further discuss and compare the two transformations via an illustrative example in Section 5.1.

Our transformation might be viewed as a way of transforming protocols into fail-stop protocols, introduced by Gong and Syverson [GS95], where any interference of an attacker is immediately observed and causes the execution to stop. But for fail-stop protocols, it is still necessary to consider the security issues related to the presence of passive adversaries. Here we achieve more since we obtain directly secure protocols. Moreover, a major difference is that we provide formal proof of the security of the resulting protocols while the approach of [GS95] is rather a methodology for prudent engineering. In particular, there are no proved guarantees on the security of the resulting protocols.

Corin *et al* [CDF+07] present a compiler for sessions (seen as patterns of communication) given as type declarations in an extended ML language to security protocols implementing the sessions. They prove that the resulting protocol implementation guarantees session integrity (which can be expressed as a set of correspodance properties). Their work can be seen as complementary with ours, since, while sesions are more general than the protocols that we consider (sessions can include loops), their compiler does not consider confidentiality properties.

Outline of the chapter Section 5.1 contains an example which illustrates the differences between the compiler of Katz and Yung and our compiler. In Section 5.2 we present the model in which we reason about security protocols. Section 5.3 introduces a simple logic and defines security properties within this logic. The protocol transformation is presented in Section 5.4. In Section 5.5 we present our main transfer result and sketch its proof, while in Section 5.5.6 we give detailed proofs.

5.1 Comparison with Katz and Yung's compiler

Consider the following simple protocol where an agent A sends a session key K_{ab} to B using his public key. Then B acknowledges A's message by forwarding the session key, encrypted under A's public key. We say that this protocol is secure if it preserves the secrecy of K_{ab}.

$$\begin{array}{ll} A \to B: & \{\!|K_{ab}|\!\}_{\mathsf{ek}(B)} \\ B \to A: & \{\!|K_{ab}|\!\}_{\mathsf{ek}(A)} \end{array}$$

Note that this protocol is secure when there is no adversary and is also secure even in the presence of an eavesdropper that may read any message sent over the network but cannot interfere in the protocol.

The resulting protocol obtained after applying Katz and Yung's compiler is the following one.

$$\begin{array}{ll} A \to B: & A, N_a \\ B \to A: & B, N_b \\ A \to B: & [\![\{\!|K_{ab}|\!\}_{\mathsf{ek}(B)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(A)} \\ B \to A: & [\![\{\!|K_{ab}|\!\}_{\mathsf{ek}(A)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(B)} \end{array}$$

However, the compiled protocol is not secure against an adversary that may use corrupted identities. Note that the message $[\![\{\!|K_{ab}|\!\}_{\mathsf{ek}(B)}]\!]_{\mathsf{sk}(A)}$ entirely reveals the message $\{\!|K_{ab}|\!\}_{\mathsf{ek}(B)}$. We assume that the adversary owns a corrupted identity I. The attack works as follows.

$$\begin{array}{llcll}
(1).1 & A & \rightarrow & B: & A, N_a \\
(1).2 & B & \rightarrow & A: & B, N_b \\
(1).3 & A & \rightarrow & B: & [\![\{\![K_{ab}]\!\}_{\mathsf{ek}(B)}, A, B, N_a, N_b]\!]_{\mathsf{sk}(A)} \\
(2).1 & I & \rightarrow & B: & I, N_i \\
(2).2 & B & \rightarrow & I: & B, N_b' \\
(2).3 & I & \rightarrow & B: & [\![\{\![K_{ab}]\!\}_{\mathsf{ek}(B)}, I, B, N_i, N_b']\!]_{\mathsf{sk}(I)} \\
(2).4 & B & \rightarrow & I: & [\![\{\![K_{ab}]\!\}_{\mathsf{ek}(I)}, I, B, N_i, N_b']\!]_{\mathsf{sk}(B)}
\end{array}$$

This allows the intruder to learn any session key used between two honest agents.

In contrast, after applying our own transformation, the resulting protocol would be secure for an unbounded number of sessions, against a fully active attacker.

5.2 The model

In this chapter we basically use the model presented in Chapter 1. We consider again only roles with matching and we will just call them roles. In this section we introduce some useful notations and we detail further the model presented in Section 1.4.2.

The concrete setting (that is, the signature, deduction system) are almost as in Chapter 2. That is, the sort system is Sorts_0 (with all sorts being different). The function symbols used here are those occurring in the deduction system $\mathcal{I}_0$ presented in Section 1.2.3.1 (page 40), and $\mathcal{I}_0$ is also the deduction system we consider in this chapter. Again the presence (or absence) of the rule (Retr) is not relevant here.

We denote by $\mathcal{X}.\mathsf{a}, \mathcal{X}.\mathsf{n}, \mathcal{X}.\mathsf{k}$ be sets of variables of sort agent, nonce, symmetric key. Variables are represented by capital letters X, A, N, K.

Throughout the chapter we fix a constant $k \in \mathbb{N}$ that represents the number of protocol participants. Furthermore, without loss of generality, we only use the set of agent variables $\{A_1, A_2, \ldots, A_k\} \subseteq \mathcal{X}.\mathsf{a}$, and we partition the set of nonce (and key) variables, according to the party that generates them. Formally:

$$\begin{array}{l}
\mathcal{X}.\mathsf{n} = \cup_{A \in \mathcal{X}.\mathsf{a}} \mathcal{X}_\mathsf{n}(A) \text{ where } \mathcal{X}_\mathsf{n}(A) = \{N_A^j \mid j \in \mathbb{N}\} \\
\mathcal{X}.\mathsf{k} = \cup_{A \in \mathcal{X}.\mathsf{a}} \mathcal{X}_\mathsf{k}(A) \text{ where } \mathcal{X}_\mathsf{k}(A) = \{K_A^j \mid j \in \mathbb{N}\}
\end{array}$$

This partition avoids to have to specify later which of the nonces (symmetric keys) are generated by the party executing the protocol, or are expected to be received from other parties.

In the same spirit, we define the following private constants of sort SymKey and Nonce respectively:

$$\begin{array}{ll}
\mathsf{T}_{\mathsf{SymKey}} & = \{\mathsf{k}^{a,j,s} \mid a \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}}), j \in \mathbb{N}, s \in \mathbb{N}\} \\
\mathsf{T}_{\mathsf{Nonce}} & = \{\mathsf{n}^{a,j,s} \mid a \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}}), j \in \mathbb{N}, s \in \mathbb{N}\}
\end{array}$$

Let $\Pi = (\mathcal{R}, \mathcal{S})$ be a protocol with k participants. For each role r of Π, we suppose that its parameters are $A_1, \ldots, A_k$ and its fresh items are among $N_{A_r}^j$ and $K_{A_r}^j$ with $j \in \mathbb{N}$. We recall that the principal that executes role $\mathcal{R}(r)$ is represented by the parameter A_r, thus, in that role, every variable of the form $X_{A_r}^j$ represents a nonce or a symmetric key generated by A_r. In this way, it is not necessary anymore to specify the parameters and fresh items of a role. Hence, we denote the r-th role of Π by $\mathcal{R}(r) = ((\mathsf{rcv}_r^1, \mathsf{snt}_r^1), (\mathsf{rcv}_r^2, \mathsf{snt}_r^2), \ldots)$, where rcv_r^p and snt_r^p are the "receive" and respectively the "send" terms of role r at step p.

Exemple 5.1 *Using the above conventions, the Needham-Schroeder protocol (presented in Section 0.1.2, page 12) is now specified by*

$$\begin{array}{lll} \mathcal{R}(1): & (\mathsf{init},\ \{\![N^1_{A_1}, A_1]\!\}_{\mathsf{ek}(A_2)}) & \mathcal{S}(1,1)=(0,0) \\ & (\{\![N^1_{A_1}, N^1_{A_2}]\!\}_{\mathsf{ek}(A_1)},\ \{\![N^1_{A_2}]\!\}_{\mathsf{ek}(A_2)}) & \mathcal{S}(1,2)=(2,1) \\ \mathcal{R}(2): & (\{\![N^1_{A_1}, A_1]\!\}_{\mathsf{ek}(A_2)},\ \{\![N^1_{A_1}, N^1_{A_2}]\!\}_{\mathsf{ek}(A_1)}) & \mathcal{S}(2,1)=(1,1) \\ & (\{\![N^1_{A_2}]\!\}_{\mathsf{ek}(A_2)},\ \mathsf{stop}) & \mathcal{S}(2,2)=(1,2) \end{array}$$

Here the notations are overloaded: for example, $N^1_{A_1}$ denotes a fresh item of sort Nonce *in role $\mathcal{R}(1)$, while it is just an arbitrary variable of sort* Msg *in role $\mathcal{R}(2)$.*

In this chapter we are also more explicit about corrupted agents and the initial knowledge of the intruder. In Chapter 1 (see Section 1.4.2, page 48) we have supposed that the corruption of agents is implicit, assuming that the data he obtained in this way is present in his initial knowledge (which was considered arbitrary). Here we assume that corrupting an agent is an explicit action of the intruder and that the initial knowledge H_0 is such that it does not contain any agent private data. This is only because we need later to differentiate between honest and corrupted agents. These extensions of the model (which are formalised next) only detail it, but do not restrict, nor generalise the class of protocols that we treat, and do not change their semantics.

A trace $\mathsf{tr} = (\mathsf{SId}_0, \mathsf{f}_0, \mathsf{H}_0) \xrightarrow{\alpha_1} (\mathsf{SId}_1, \mathsf{f}_1, \mathsf{H}_1) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ is defined by the statement of Definition 1.29 and by:

- H_0 is such that $\mathsf{H}_0 \not\vdash \mathsf{k}(a,b), \mathsf{dk}(a), \mathsf{sk}(a)$, for all $a, b \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F})$,
- α_1 can also be the **corrupt** action: $(\mathsf{SId}_0, \mathsf{f}_0, H_0) \xrightarrow{\mathbf{corrupt}(a_1,\ldots,a_l)} (\mathsf{SId}_1, \mathsf{f}_1, \mathsf{H}_1)$ where $a_1, \ldots, a_l \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}})$ and $\mathsf{H}_1 = \mathsf{H}_0 \cup \bigcup_{1\le j\le l} (\{\mathsf{dk}(a_j), \mathsf{sk}(a_j)\} \cup SK(a_j))$. Here, $SK(a)$ denotes a finite set of symmetric keys shared by the agent a with other agents, that is $SK(a) \subseteq \{\mathsf{k}(a,b), \mathsf{k}(b,a) \mid b \in \mathcal{T}_{\mathsf{Id}}(\mathcal{F}_{\mathsf{pub}})\}$.
- if $\alpha_i = \mathbf{new}(r, a_1, \ldots, a_k)$ and $\mathsf{sid} = (s, r, (a_1, \ldots, a_k))$ is the new session id then $\mathsf{f}_i(\mathsf{sid}) = (\sigma, p_0)$ where p_0 is the initial control point of role[17] r, and

$$\begin{cases} \sigma(A_j) &= a_j & 1 \le j \le k \\ \sigma(N^j_{A_r}) &= \mathsf{n}^{a_r,j,s} & N^j_{A_r} \text{ fresh item of role } \mathcal{R}(r) \\ \sigma(K^j_{A_r}) &= \mathsf{k}^{a_r,j,s} & K^j_{A_r} \text{ fresh item of role } \mathcal{R}(r) \end{cases}$$

Given a protocol Π, we write $\mathsf{Exec}(\Pi)$ for the set of execution traces of Π. When specifying a trace, we sometimes omit the transitions and write it just as a sequence of states.

Exemple 5.2 *Reexamining Example 1.30 (page 50), we obtained the following execution trace:*

$$\begin{array}{l} (\emptyset, f_1, \emptyset) \xrightarrow{\mathbf{corrupt}(a_3)} (\emptyset, f_1, \mathsf{kn}) \xrightarrow{\mathbf{new}(2,a_1,a_2)} (\{\mathsf{sid}_1\}, f_2, \mathsf{kn}) \\ \qquad\qquad\qquad \xrightarrow{\mathbf{send}(\mathsf{sid}_1, \{\![\mathsf{n}_3, a_1]\!\}_{\mathsf{ek}(a_2)})} (\{\mathsf{sid}_1\}, f_3, \mathsf{kn} \cup \{\![\mathsf{n}_3, \mathsf{n}^{a_2,1,1}]\!\}_{\mathsf{ek}(a_1)}), \end{array}$$

where $\mathsf{kn} = \{\mathsf{dk}(a_3), \mathsf{sk}(a_3)\}$, $\mathsf{sid}_1 = (1, 2, (a_1, a_2))$, and f_2, f_3 are defined as follows: $f_2(\mathsf{sid}_1) = (\sigma_1, 2, 1)$, $f_3(\mathsf{sid}_1) = (\sigma_2, 2, 2)$ where $\sigma_1(A_1) = a_1$, $\sigma_1(A_2) = a_2$, $\sigma_1(N^1_{A_2}) = \mathsf{n}^{a_2,1,1}$, and σ_2 extends σ_1 by $\sigma_2(N^1_{A_1}) = \mathsf{n}_3$, with a_1, a_2, a_3 public constants of sort Id*, and $\mathsf{n}^{a_2,1,1}, \mathsf{n}_3$ (private, respectively public) constants of sort* Nonce.

[17] The initial control point p_0 is usually 1, but for technical reasons here it may also be some other integer.

In this chapter we consider a relaxed version of the definition of executable protocols. Indeed, for technical reasons we work only with protocols satisfying the first two points of the Definition 1.31 (on page 50) and we call such protocols executable. In particular, we suppose that the function $\mathcal{S}$ is injective. Moreover, since we sometimes use negative control points (with negatively indexed role rules), we consider that for executable protocols the function $\mathcal{S}(r, \cdot)$ is defined on exactly $\sharp\mathcal{R}(r)$ consecutive integers.

Given an arbitrary trace $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ with $n \in \mathbb{N}$, we define the set of *corrupted agents* of a trace tr by $\mathsf{CA}(\mathsf{tr}) = \{a_1, \ldots, a_l\}$ if $\alpha_1 = \mathbf{corrupt}(a_1, \ldots, a_l)$ and $\mathsf{CA}(\mathsf{tr}) = \emptyset$ otherwise. The set $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ of honest session identifiers is the set of session identifiers that correspond to sessions between non-corrupt agents:

$$\mathsf{SId}^{\mathsf{h}}(\mathsf{tr}) = \{\mathsf{sid} \in \mathsf{SId}_n \mid \mathsf{sid} = (\cdot, \cdot, (a_1, \ldots, a_k)), \mathsf{CA}(\mathsf{tr}) \cap \{a_1, \ldots, a_k\} = \emptyset\}.$$

For a trace tr we denote by $\mathsf{I}(\mathsf{tr})$ the set of indexes i of the transitions and global states of tr. For example, the above trace has $\mathsf{I}(\mathsf{tr}) = \{0, 1, \ldots, n\}$. If sid is a session id then we denote by $\mathsf{Ag}(\mathsf{sid})$ the set of agents involved in this session, that is $\mathsf{Ag}(\mathsf{sid}) = \{a_1, \ldots, a_k\}$ when $\mathsf{sid} = (\cdot, \cdot, (a_1, \ldots, a_k))$.

5.3 Security properties

We use a simple logic (similar with the one in [CHW06]) to express security properties on traces. We define the syntax and semantics of this logic and provide several examples of security properties that can be expressed within it.

5.3.1 A logic for security properties

5.3.1.1 Syntax

We assume an infinite set $\mathcal{X}_{Sub}$ of variables for substitutions, called *substitution variables*. Let T_{Sub} be the following set inductively defined by:

$$\mathsf{T}_{Sub} \ ::= \ \varsigma(X) \mid c \mid g(\mathsf{T}_{Sub}) \mid h(\mathsf{T}_{Sub}, \mathsf{T}_{Sub})$$

where $\varsigma \in \mathcal{X}_{Sub}$, $X \in \mathcal{X}$, and c, g, h are function symbols of arity 0, 1 and 2 respectively that range over *Sigma*. We call *s-terms* the elements of T_{Sub}. Note that terms without names are s-terms, and s-terms without substitution variables are terms. We extend the notions of position and occurrence to s-terms in the expected way (with ς regarded as a function symbol of arity 1). If u is an s-term and σ is a substitution then we denote by $u[{}^{\sigma}/_{\varsigma}]$ the s-term obtained by replacing each occurrence of $\varsigma(X)$ by the term $\sigma(X)$, for any variable X. As for normal substitutions, we may abbreviate $\varsigma(X)$ by $X\varsigma$.

Exemple 5.3 *Let $u = \langle \varsigma(X), \varsigma'(Y) \rangle$ and σ, σ' be two substitutions such that $X\sigma = a$ and $Y\sigma' = b$. Let $u' = u[{}^{\sigma}/_{\varsigma}] = \langle \sigma(X), \varsigma'(Y) \rangle = \langle a, \varsigma'(Y) \rangle$. Then $u'[{}^{\sigma'}/_{\varsigma'}] = \langle a, b \rangle$ is a both a s-term and a term, while u' is a s-term but not a term.*

Besides standard propositional connectors, the logic has equality tests between s-terms, a predicate to specify honest agents, and existential and universal quantifiers over the local states of agents.

$$
[\![\mathsf{NC}(a), \mathsf{tr}]\!] = \begin{cases} 1 & \text{if } a \notin \mathsf{CA}(\mathsf{tr}) \\ 0 & \text{otherwise} \end{cases}
$$

$$
[\![\forall \varsigma \in \mathcal{LS}_{r,p}\ \phi, \mathsf{tr}]\!] = \begin{cases} 1 & \text{if } \forall \sigma \in \mathcal{LS}_{r,p}(\mathsf{tr}), \text{ we have } [\![\phi[{}^{\sigma}/_{\varsigma}], \mathsf{tr}]\!] = 1, \\ 0 & \text{otherwise.} \end{cases}
$$

$$
[\![\exists \varsigma \in \mathcal{LS}_{r,p}\ \phi, \mathsf{tr}]\!] = \begin{cases} 1 & \text{if } \exists \sigma \in \mathcal{LS}_{r,p}(\mathsf{tr}), \text{ s.t. } [\![\phi[{}^{\sigma}/_{\varsigma}], \mathsf{tr}]\!] = 1, \\ 0 & \text{otherwise.} \end{cases}
$$

$$
[\![\exists! \varsigma \in \mathcal{LS}_{r,p}\ \phi, \mathsf{tr}]\!] = \begin{cases} 1 & \text{if } \exists!\, \mathsf{sid} \in \mathsf{SId}(\mathsf{tr}), \exists i \in \mathsf{I}(\mathsf{tr}) \text{ s.t.} \\ & \qquad f_i(\mathsf{sid}) = (\sigma, p) \text{ and } [\![\phi[{}^{\sigma}/_{\varsigma}], \mathsf{tr}]\!] = 1, \\ 0 & \text{otherwise.} \end{cases}
$$

Figure 5.1: Interpretation of formulas in $\mathcal{L}$.

Definition 5.4 *The formulas of the logic $\mathcal{L}$ are inductively defined by:*

$$
\phi \ ::= \ \neg\phi \mid \phi \wedge \phi \mid [u = v] \mid \mathsf{NC}(\varsigma(A)) \mid \forall \varsigma \in \mathcal{LS}_{r,p}\ \phi \mid \exists \varsigma \in \mathcal{LS}_{r,p}\ \phi \mid \exists! \varsigma \in \mathcal{LS}_{r,p}\ \phi
$$

where $A \in \mathcal{X}.a$, $\varsigma \in \mathcal{X}_{Sub}$, $u, v \in \mathsf{T}_{Sub}$ and $r, p \in \mathbb{N}$.

Here the predicate $\mathsf{NC}(\varsigma(A))$ of arity 1 is used to specify non corrupted agents. The quantifications $\forall \varsigma \in \mathcal{LS}_{r,p}$, $\exists \varsigma \in \mathcal{LS}_{r,p}$, and $\exists! \varsigma \in \mathcal{LS}_{r,p}$ are over local states of agent r at step p, as defined below. All these quantifiers bound the substitution variable ς. Hence, a formula ϕ of $\mathcal{L}$ is *closed* if all substitution variables in ϕ are bound. For a formula ϕ and a substitution σ, the formula $\phi[{}^{\sigma}/_{\varsigma}]$ is defined as expected, by replacing each s-term u in ϕ by $u[{}^{\sigma}/_{\varsigma}]$.

As usual, we assume that unary predicates bind tighter than binary predicates and the precedence for the latter predicates is $\Rightarrow, \vee, \wedge$ (in ascending order).

5.3.1.2 Semantics

The semantics of our logic is defined for closed formula by interpreting them on the execution traces of a protocol. We first define formally the range of the quantifications. The set of *local states* of role r at step p in a trace $\mathsf{tr} = (\mathsf{SId}_i, f_i, H_i)_{1 \leq i \leq n}$ is defined by

$$
\mathcal{LS}_{r,p}(\mathsf{tr}) = \{\sigma \mid \exists i \in [n], \exists \mathsf{sid} \in \mathsf{SId}_i \text{ s.t. } f_i(\mathsf{sid}) = (\sigma, p)\}.
$$

Standard propositional connectors and negation are interpreted as usual. Equality is syntactic equality between terms, that is, $[\![[u = v], \mathsf{tr}]\!] = 1$ if and only if $u = v$ (the interpretation is thus trace-independent). Indeed, when $[u = v]$ is a closed formula, the s-terms u and v are terms. Note that the terms u, v need not be ground. The interpretation of quantifiers and the predicate NC is shown in Figure 5.1.

As usual, we use $[u \neq v]$, $\phi_1 \vee \phi_2$ and $\phi_1 \Rightarrow \phi_2$ as shortcuts for $\neg[u = v]$, $\neg(\neg\phi_1 \wedge \neg\phi_2)$, and $\neg\phi_1 \vee \phi_2$ respectively.

A security property is represented by a closed formula in the logic $\mathcal{L}$. Informally, a protocol Π satisfies ϕ if ϕ is true on all traces of Π. Formally:

Definition 5.5 (Satisfiability) *Let Π be a protocol and $\phi \in \mathcal{L}$ be a closed formula. We say that* Π satisfies ϕ, *and write* $\Pi \models \phi$ *if for any trace* $\mathsf{tr} \in \mathsf{Exec}(\Pi)$, $[\![\phi, \mathsf{tr}]\!] = 1$.

5.3.2 Examples of security properties

In this section we show how to specify secrecy and several variants of authentication, including those from Lowe's hierarchy [Low97], in the given security logic.

5.3.2.1 Secrecy

Let Π be a k-party executable protocol. To specify our secrecy property we use a standard encoding. Namely, we add a role to the protocol, $\mathcal{R}(k+1) = (Y, \mathsf{stop})$, where Y is a new variable of sort Msg. It can be seen as some sort of witness as it does nothing but waits for receiving a public data. Then a data s is secret if and only if for any role session of role $k+1$, the value of s is different from the value of Y.

Consider X a fresh item of a role r. Informally, the definition of the secrecy property expressed by the formulas ϕ_s below states that, for any local state of an agent playing role r in an honest session, a witness (i.e. an agent playing role $k+1$) cannot gain any knowledge on X. Formally, the property is specified by the following formula:

$$\phi_s \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{r,1} \Big(\bigwedge_{l \in [k]} NC(\varsigma(A_l)) \Rightarrow \forall \varsigma' \in \mathcal{LS}_{k+1,2} \left[\varsigma(X) \neq \varsigma'(Y)\right] \Big)$$

Note that, due to the assumption that X is a fresh item and the role $k+1$ is at its final control point, the interpreted s-terms $\varsigma(X)$ and $\varsigma'(Y)$ are ground terms. We can also model the secrecy of a data X that is received in an honest session: we simply specify the control point p (instead of 1) at which the data is received by the role r.

5.3.2.2 Authentication properties

We first show how to use the logic defined above to specify the injective agreement [Low97] between two parties A and B. Informally, this property states that whenever an agent A completes a run of the protocol, apparently with B, then there is unique run of B apparently with A such that two agents agree on the values of some fixed data items[18] $\{X_1, \dots, X_n\}$, provided that A and B are honest. As usual, nothing is guaranteed in role sessions involving corrupted agents.

Let p_1 be the length of A's role, p_2 be the control point at which B should have received all data items from A, and assume the indexes of A's and B's roles are 1 and 2 respectively. Then, the above intuition is captured by the following formula:

$$\phi_a \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{1,p_1} \Big(\mathsf{NC}(A\varsigma) \wedge \mathsf{NC}(B\varsigma) \Rightarrow \exists! \varsigma' \in \mathcal{LS}_{2,p_2} \big([A\varsigma = A\varsigma'] \wedge [B\varsigma = B\varsigma'] \wedge \bigwedge_{1 \leq i \leq n} [X_i\varsigma = X_i\varsigma']\big)\Big)$$

We show next how other several authentication definitions proposed by Lowe [Low97] can be modelled within the logic $\mathcal{L}$. The following formulas represent aliveness, weak agreement, and non-injective agreement properties, respectively, where A, B, p_1, p_2 and $\{X_1, \dots, X_n\}$ have the same meaning as above.

[18] We assume that these data items are represented by variables with the same name respectively in the two roles; the formula can be easily changed when the data items are represented by arbitrary terms.

$$\phi_1 \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{1,p_1+1} \left(\mathsf{NC}(A\varsigma) \wedge \mathsf{NC}(B\varsigma) \Rightarrow \exists \varsigma' \in \mathcal{LS}_{2,1} \left[B\varsigma = B\varsigma'\right]\right)$$

$$\phi_2 \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{1,p_1+1} \Big(\mathsf{NC}(A\varsigma) \wedge \mathsf{NC}(B\varsigma) \Rightarrow \exists \varsigma' \in \mathcal{LS}_{2,1} \left(\left[B\varsigma = B\varsigma'\right] \wedge \left[A\varsigma = A\varsigma'\right]\right)\Big)$$

$$\phi_3 \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{1,p_1+1} \Big(\mathsf{NC}(A\varsigma) \wedge \mathsf{NC}(B\varsigma) \Rightarrow \exists \varsigma' \in \mathcal{LS}_{2,p_2} \big(\left[B\varsigma = B\varsigma'\right] \wedge \left[A\varsigma = A\varsigma'\right] \wedge \bigwedge_{1 \le i \le n} \left[X_i\varsigma = X_i\varsigma'\right]\big)\Big)$$

We also model a simple security property for the multi-party case by requiring that each party authenticates any other party in the sense that each agent is convinced that the other agents were alive in the session. In our logic, this translates to the formula: $\phi_{ma} \stackrel{\text{def}}{=} \bigwedge_{r \in [k]} \phi_{ma}(r)$ with

$$\phi_{ma}(r) \stackrel{\text{def}}{=} \forall \varsigma \in \mathcal{LS}_{r,p_r+1} \Big(\bigwedge_{l \in [k]} \mathsf{NC}(A_l\varsigma) \Rightarrow \bigwedge_{i \in [k], i \neq r} \big(\exists \varsigma_i \in \mathcal{LS}_{i,1} \bigwedge_{j \in [k]} \left[A_j\varsigma = A_j\varsigma_i\right]\big)\Big)$$

where p_r is the final control point of the role r. We could enforce the property as for the two-party case by enlarging the set of equalities that should hold.

5.4 Transformation of protocols

The core idea of the transformation is to have parties agree on some common, dynamically generated, session identifier s, and then transmit the encryption of a message m of the original protocol accompanied by a signature on m concatenated with s.

The modification of the source protocol is performed in two steps. We first introduce an initialisation phase, where each agent generates a fresh nonce which is distributed to all other participants. The idea is that the concatenation of all these nonces and all the identities involved in the session plays the role of a unique session identifier. To avoid underspecification of the resulting protocol we fix a particular way in which the nonces are distributed. First, each agent generates a fresh nonce and then sends the nonces he received so far together with his nonce to the next agent. That is, in Alice-Bob notation,

$$A_i \rightarrow A_{i+1} : \; N_{A_1}, \dots, N_{A_i}$$

for all i in the sequence $1, \dots, k-1$. Then, once the last agent received all nonces, each agent forwards the concatenation of all nonces to its predecesor. That is,

$$A_i \rightarrow A_{i-1} : \; N_{A_1}, \dots, N_{A_k}$$

for all i in the sequence $k, \dots, 2$. In this way, at the end of this first phase all agents know each other's nonces.

We remark that the precise order in which participants send these nonces does not really matter, and we do not require that these nonces be authenticated in some way. In principle an active adversary is allowed to forward, block or modify the messages sent during the initialisation phase, but behaviours that deviate from the intended execution of the protocol are detected in the next phase.

In the second phase of the transformed protocol, the execution proceeds as prescribed by Π with the difference that to each message m that needs to be sent, the sending parties also attaches a signature $[\![m, p, \mathsf{nonces}]\!]_{\mathsf{sk}'(a)}$ and encrypts the whole construct with the intended receiver public key. p is the current control point and nonces is the concatenation of the nonces received during the first phase with the identities of the participants involved in the protocol. To avoid confusion and unintended interactions between the signatures and the encryptions produced by the compiler and those used in the normal execution of the protocol, the former use fresh signatures and public keys. Formally, we extend the signature Σ with four new function symbols sk', vk', ek' and dk' which have exactly the same functionality (that is the same sort and similar deduction rules) with sk, vk, ek and dk respectively. This formalises the assumption that in the transformed version of Π each agent a has associated two pairs of verification/signing keys $((\mathsf{vk}(a), \mathsf{sk}(a))$ and $(\mathsf{vk}'(a), \mathsf{sk}'(a)))$ and two pairs of encryption/decryption keys $((\mathsf{ek}(a), \mathsf{dk}(a))$ and $(\mathsf{ek}'(a), \mathsf{dk}'(a)))$ and that these new pairs of keys were correctly distributed previously to any execution of the protocol. We assume that source protocols are constructed over Σ only.

Definition 5.6 (Transformed protocol) *Let $\Pi = (\mathcal{R}, \mathcal{S})$ be a k-party executable protocol such that the nonce variables $N^0_{A_i}$ do not appear in $\mathcal{R}$ (which can be ensured by renaming the nonce variables of Π) and all the initial control points are set to 1 (which can be ensured by rewriting the function $\mathcal{S}$).*

The transformed protocol $\widetilde{\Pi} = (\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$ *is defined as follows: $\widetilde{\mathcal{R}}(r) = \mathcal{R}^{\mathsf{init}}(r) \cdot \mathcal{R}'(r)$ and $\widetilde{\mathcal{S}} = \mathcal{S}^{\mathsf{init}} \cup \mathcal{S}$ where $\cdot$ denotes the concatenation of sequences and $\mathcal{R}^{\mathsf{init}}$, $\mathcal{R}'$ and $\mathcal{S}^{\mathsf{init}}$ are defined as follows:*

$$\begin{array}{l} \mathcal{R}^{\mathsf{init}}(r) = \big((\mathsf{nonces}_{r-1}, \mathsf{nonces}_r), (\mathsf{nonces}_k, \mathsf{nonces}_k)\big), \quad \forall 1 \leq r < k, \\ \mathcal{S}^{\mathsf{init}}(r, -1) = (r-1, -1), \quad \mathcal{S}^{\mathsf{init}}(r, 0) = (r+1, 0), \quad \forall 1 \leq r < k, \\ \mathcal{R}^{\mathsf{init}}(k) = \big((\mathsf{nonces}_{k-1}, \mathsf{nonces}_k)\big) \qquad \mathcal{S}^{\mathsf{init}}(k, 0) = (k-1, -1) \end{array}$$

with $\mathsf{nonces}_0 = \mathsf{init}$ *and* $\mathsf{nonces}_j = \langle N^0_{A_1}, N^0_{A_2}, \ldots, N^0_{A_j} \rangle$ *for* $1 \leq j \leq k$.
Let $\mathcal{R}(r) = \big((\mathsf{rcv}^p_r, \mathsf{snt}^p_r)\big)_{p \in [k_r]}$. *Then* $\mathcal{R}'(r) = \big((\widetilde{\mathsf{rcv}}^p_r, \widetilde{\mathsf{snt}}^p_r)\big)_{p \in [k_r]}$ *such that if* $\mathsf{rcv}^p_r = \mathsf{init}$ *then* $\widetilde{\mathsf{rcv}}^p_r = \mathsf{fake}$, *if* $\mathsf{snt}^p_r = \mathsf{stop}$ *then* $\widetilde{\mathsf{snt}}^p_r = \mathsf{stop}$ *and otherwise*

$$\begin{array}{l} \widetilde{\mathsf{rcv}}^p_r = \{\![\mathsf{rcv}^p_r, [\![\mathsf{rcv}^p_r, p', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r'})}]\!\}_{\mathsf{ek}'(A_r)}, \\ \widetilde{\mathsf{snt}}^p_r = \{\![\mathsf{snt}^p_r, [\![\mathsf{snt}^p_r, p, \mathsf{nonces}]\!]_{\mathsf{sk}'(A_r)}]\!\}_{\mathsf{ek}'(A_{r''})} \end{array}$$

where $(r', p') = \mathcal{S}(r, p)$, $(r, p) = \mathcal{S}(r'', p'')$ *and* $\mathsf{nonces} = \langle A_1, \ldots, A_k, \mathsf{nonces}_k \rangle$.

The initial control point is now set to -1 (or 0 for A_k) since actions have been added for the initialisation stage. The special message fake is used to model for example the situation where an agent waits for more than one message in order to reply or when an agent sends more then one reply.

5.5 Transfer result

5.5.1 Honest, single session traces

We identify a class of executions, which we call honest, single session executions which, intuitively, correspond to traces where just one session is executed, session in which all parties are honest and there is no adversary. Our only hypothesis will be that the initial protocol has to be secure in this very weak setting.

Definition 5.7 (Honest, single session trace) *Let* $\Pi = (\mathcal{R}, \mathcal{S})$ *be a* k*-party protocol and* $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ *be an execution trace of* Π*. The trace* tr *is an* honest, single session trace *if there are* k *agent identities* $a_1, \ldots, a_k$ *such that*

- *for* $1 \leq i \leq k$*,* $\alpha_i = \mathbf{new}(i, a_1, \ldots, a_k)$*,*
- *for* $k+1 \leq i \leq n$*,* $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$*,* $m = \mathsf{rcv}_r^p \sigma$ *where* $f_i(\mathsf{sid}) = (\sigma, p+1)$*,* $\mathsf{sid} = (\cdot, r, \cdot)$*, and there exists* $j < i$ *such that* $f_j(\mathsf{sid}') = (\sigma', p')$*,* $\mathcal{S}(r, p) = (r', p')$*,* $\mathsf{sid}' = (\cdot, r', \cdot)$*, and* $m = \mathsf{snt}_{r'}^{p'} \sigma'$ *for some* sid'*.*

We denote by $\mathsf{Exec}^{p,1}(\Pi)$ the set of honest, single session traces of Π.

Definition 5.8 (Passive, single session satisfiability) *Let* Π *be a protocol and* $\phi \in \mathcal{L}$ *be a closed formula. We say that* Π *satisfies the closed formula* ϕ *for* passive adversaries and a single session*, and write* $\Pi \models^{p,1} \phi$ *if for any trace* $\mathsf{tr} \in \mathsf{Exec}^{p,1}(\Pi)$*,* $[\![\phi, \mathsf{tr}]\!] = 1$*.*

5.5.2 Transferable security properties

We identify a fragment $\mathcal{L}'$ of the logic $\mathcal{L}$ defined in Section 5.3 whose formulas specify the properties that can be transferred from the honest, single session case to the full active adversary case.

Definition 5.9 *The set* $\mathcal{L}'$ *consists of those formulas* ϕ *with*

$$\phi = \forall \varsigma \in \mathcal{LS}_{r,p} \Big(\bigwedge_{l \in [k]} \mathsf{NC}(A_l \varsigma) \Rightarrow \bigwedge_{i \in I} \big(\mathcal{Q}_i \, \varsigma_i \in \mathcal{LS}_{r_i, p_i} \bigwedge_{j \in J_i} \tau_j^i (u_j^i, v_j^i) \big) \Big)$$

where $\mathcal{Q}_i \in \{\forall, \exists, \exists !\}$*, and for all* $i \in I$*, for all* $j \in J_i$*, if* $\mathcal{Q}_i = \forall$ *then* $\tau_j^i \in \{\neq\}$ *and if* $\mathcal{Q}_i \in \{\exists, \exists !\}$ *then* $\tau_j^i \in \{=, \neq\}$*; moreover, for each* $i \in I$*, if* $\mathcal{Q}_i = \forall$ *(respectively* $\mathcal{Q}_i = \exists !$*) then for all (there is)* $j \in J_i$ *we have that (such that* $\tau_j^i \in \{=\}$ *and) there exists at least a subterm* $\varsigma(X)$ *in* u_j^i *or* v_j^i *with* X *a nonce or key variable.*

As usual, we require security properties to hold in sessions between honest agents. This means that no guarantee is provided in a session where a corrupted agent is involved. But this does not prevent honest agents from contacting corrupted agents in parallel sessions. Properties that can be expressed in our fragment $\mathcal{L}'$ are correspondence relations between (data in) particular local states of agents in different sessions. It is a non-trivial class since e.g. the logical formulas given in Section 5.3 for expressing secrecy and authentication are captured by the above definition.

5.5.3 Transference theorem

The main result of this chapter is the following transference theorem. It informally states that the formulas of $\mathcal{L}'$ that are satisfied in single, honest executions of a protocol are also satisfied by executions of the transformed protocol in the presence of a fully active adversary.

Theorem 5.10 *Let* Π *be a protocol and* $\widetilde{\Pi}$ *the corresponding transformed protocol. Let* $\phi \in \mathcal{L}'$ *be a closed formula. Then* $\Pi \models^{p,1} \phi \Rightarrow \widetilde{\Pi} \models \phi$*.*

Remark The transfer result holds in fact for any conjunction or disjunction of formulas in $\mathcal{L}'$. Indeed, if $\phi_1 \wedge \phi_2$ is satisfied by a protocol Π then both ϕ_1 and ϕ_2 are satisfied by Π. Then applying twice the transfer result, once for each formula, we obtain that $\phi_1 \wedge \phi_2$ is also satisfied by the transformed protocol. For example, the formula ϕ_{ma} does not belong do the class $\mathcal{L}'$ (it is not of the right form). It is however a conjunction of formulas of $\mathcal{L}'$. We can thus also transfer this property.

The main intuition behind the proof is that any execution in the presence of an active adversary is closely mirrored by some *honest* execution (i.e. an execution with no adversarial interference plus some additional useless sessions). We define honest executions next.

5.5.4 Honest executions

Recall that we demand that protocols come with an intended execution order, in which the designer specifies the source of each message in an execution. Roughly, in an honest execution trace one can partition the set of session ids in sets of at most k role sessions (each corresponding to a different role of the protocol) such that messages are exchanged only within partitions, and the message transmission within each partition follows the intended execution specification. Since we cannot prevent an intruder to create new messages and sign them with corrupted signing keys, clearly the property can hold only for session identifiers corresponding to honest participants. The above ideas are captured by the following definition.

Definition 5.11 (Honest execution traces) *Let Π be an executable protocol. An execution trace* $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ *is* honest *if there is a partition* PrtSId *of the honest role session identifiers* $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ *such that:*

1. *for all* $S \in \mathsf{PrtSId}$*, for all* $\mathsf{sid}, \mathsf{sid}' \in S$ *with* $\mathsf{sid} \neq \mathsf{sid}'$ *and* $\mathsf{sid} = (s, r, (a_1, \ldots a_k))$ *and* $\mathsf{sid}' = (s', r', (a'_1, \ldots a'_k))$*, we have* $r \neq r'$*, and* $a_j = a'_j$ *for all* $1 \leq j \leq k$*; that is, in any protocol session each of the participants execute different roles*[19] *and the agents agree on their communication partners;*

2. *whenever* $(\mathsf{SId}_{i-1}, f_{i-1}, H_{i-1}) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}_i, f_i, H_i)$ *with* $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$*,* m *accepted,* $m \neq \mathsf{fake}$ *and* $m = \mathsf{rcv}_r^p\sigma$*,* $p \geq 1$*, we have that there are* $\mathsf{sid}' \in [\mathsf{sid}]$ *and* $i' < i$ *such that* $m = \mathsf{snt}_{r'}^{p'}\sigma'$ *and* $\mathcal{S}(r,p) = (r', p')$ *where* $f_i(\mathsf{sid}) = (\sigma, p+1)$*,* $f_{i'}(\mathsf{sid}') = (\sigma', p')$*,* $\mathsf{sid}' = (\cdot, r'. \cdot)$ *and* $[\mathsf{sid}]$ *is the partition to which* sid *belongs to.*

Notice that the above definition considers partial executions in which not all roles finish their execution, and where not all roles in a protocol session need to be initialised. The following lemma states that for any transformed protocol, an active intruder cannot interfere with the execution of honest sessions.

Lemma 5.12 *Let Π be a protocol and $\widetilde{\Pi}$ the corresponding transformed protocol. In $\widetilde{\Pi}$, any execution trace is an honest execution trace.*

Then it remains to show that any property expressed in $\mathcal{L}'$ that holds for one honest, single session trace also holds for any honest execution trace of the transformed protocol. It relies in particular on the fact that due to encryption, fresh values of honest sessions cannot occur unprotected in dishonest sessions. Moreover, honest execution traces actually correspond to the honest, single session trace of the initial protocol. This intuition is further detailed in the next section.

[19] Consequently, each partition consists of at most k role sessions.

5.5.5 Proof sketch of the transference theorem

In this section we first present some useful lemmas and the ideas behind their proofs, and then we sketch the proof of the main result. Detailed proofs are found in Section 5.5.6.

Lemma 5.12 *Let Π be a protocol and $\widetilde{\Pi}$ the corresponding transformed protocol. In $\widetilde{\Pi}$, any execution trace is an honest execution trace.*

Proof sketch Let tr be an execution trace of $\widetilde{\Pi}$. We construct the partition of session ids by simply grouping session ids that have the same value of nonces; we write $\mathsf{PrtSId}(\mathsf{tr})$ for the resulting partition. It is easy to check that $\mathsf{PrtSId}(\mathsf{tr})$ satisfies the first condition of the definition of an honest execution trace. We prove that the second condition also holds by induction on the length of the trace. Assume that $(\mathsf{SId}_{i-1}, f_{i-1}, H_{i-1}) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}_i, f_i, H_i)$ with $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$, m accepted, $m \neq \mathsf{init}$ and $m = \widetilde{\mathsf{rcv}}_r^p\sigma$, $p \geq 1$. Then, m must be of the form $\{\![m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b)}$ with a, b honest agents and the agents occurring in m_0 being honest too. Since the adversary cannot forge $[\![m', p, m_0]\!]_{\mathsf{sk}'(a)}$, a message of the form $\{\![m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b')}$ must have been sent by the honest agent a in a session $\mathsf{sid}' \in [\mathsf{sid}]$ since the two agents agree on m_0. Thanks to the control point that is also signed, we can show that a must have sent his message exactly to the agent that is expected, and then deduce that $b = b'$ and a's action also satisfies the condition on function $\mathcal{S}$. ∎

Any nonce or key generated in an honest session is always protected by at least one encryption with a public key of a non-corrupted agent.

Lemma 5.13 *Let Π be a k-party protocol and $\widetilde{\Pi}$ be the corresponding transformed protocol. Let X be a key or a nonce variable of Π, tr be an execution trace of $\widetilde{\Pi}$, (SId, f, H) be a global state of tr and t be a message deducible from H, i.e. $H \vdash t$.*

For any honest session id $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ with $f(\mathsf{sid}) = (\sigma, \cdot, \cdot)$, for any occurrence of $X\sigma$ in t (i.e. for any path q such that $t|_q = X\sigma$), $X\sigma$ occurs within t in messages of the form $\{\![m', [\![m', p, \sigma(\mathsf{nonces})]\!]_{\mathsf{sk}(a)}]\!\}_{\mathsf{ek}'(b)}$, where b is an honest agent, i.e. $b \notin CA(\mathsf{tr})$.

Proof sketch Using Lemma 5.12, we can show that the only possible values for $X\sigma$ are nonces generated in honest sessions. Thus $X\sigma$ is initially protected with an honest public key encryption. Then the only way for an adversary to remove that encryption is to send the message to an honest agent, which in turn will send it to one of the agents occurring in $\sigma(\mathsf{nonces})$ thus to another honest agent; still, $X\sigma$ will be protected by an honest public key encryption. ∎

5.5.5.1 Sketch of proof of the main result

Firstly, Lemma 5.12 says that it is sufficient to look at honest execution traces in the transformed protocol. So we fix an arbitrary honest trace tr. Every (possibly partial) honest protocol session in tr can be projected to a (partial) honest, single session trace tr_0 in the initial protocol. Observe that we are only interested in honest sessions of tr, since that is what the left hand side of the implication in ϕ means (i.e. $\bigwedge_{l\in[k]} \mathsf{NC}(A_l\varsigma)$). But then the hypothesis of the implication in ϕ is trivially satisfied for passive, single sessions in Π. Hence also the right hand side of the implication in ϕ is satisfied for passive, single sessions in Π.

Next we have to consider three cases according to what the quantifier $\mathcal{Q}_i$ is.

If it is $\exists$ then the there is a local state satisfying the (in)equalities simply because there is one in the honest, single session trace tr_0.

If $\mathcal{Q}_i = \exists!$ we also need to prove uniqueness. From the form of ϕ we know that there is $j \in J_i$ such that in (say) u^i_j there is an occurrence of $X\varsigma$ with X a nonce or a key variable. And from the existence of required local states we have $(u^i_j = v^i_j)[^{\sigma}/_{\varsigma}][^{\sigma'}/_{\varsigma_i}]$ for some "valid" σ and σ'. Suppose that also $(u^i_j = v^i_j)[^{\sigma}/_{\varsigma}][^{\sigma''}/_{\varsigma_i}]$ for some "valid" σ''. But then $X\sigma$ occurs both in $Y\sigma'$ and $Y\sigma''$, where Y is a variable of v^i_j such that $X\varsigma$ occurs in v^i_j under $Y\varsigma_i$. Since $Y\sigma'$ and $Y\sigma''$ are parts of sent or received messages, it follows that they are parts of messages by the intruder and hence so is $X\sigma$. Thus we can apply Lemma 5.13 and obtain that $Y\sigma'$ and $Y\sigma''$ were sent or received in the same honest protocol session as $X\sigma$. And since σ' and σ'' are substitutions obtained at the same control point p_i of the same role r_i it follows that $\sigma' = \sigma''$ hence uniqueness.

Finally, if $\mathcal{Q}_i = \forall$. This case proceeds similarly. If, by absurd, there are local states such that the terms in some test become equal for some σ' then it must be the case that the corresponding session id is honest (this is again obtained using the uniqueness of nonces created in honest sessions, that is Lemma 5.13). We can then project this equality in the honest, single session trace tr_0 to obtain a contradiction.

5.5.6 Detailed proofs

5.5.6.1 Proof of Lemma 5.12

Let $\Pi = (\mathcal{R}, \mathcal{S})$ be an executable protocol such that there are no variable nonces N^0_A in $\mathcal{R}$ and the initial control points are set to 1. Let $\mathcal{R}(r) = ((\mathsf{rcv}^p_r, \mathsf{snt}^p_r))_{1 \leq p \leq k_r}$. Consider an arbitrary execution trace tr of the protocol $\widetilde{\Pi}$. Suppose $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$. We need to show that tr is an honest execution trace.

We first give a few useful definitions and properties.

For a message m we define $\overline{m} = m_0$ if $m = \{\![m', [\![m', p, m_0]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b)}$ for some identities a, b, some messages m', m_0 and some $p \in \mathbb{Z}$ and $\overline{m} = \perp$ otherwise[20]. We call m_0 the *nonces field*[21] of m.

For any transition $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ such that m is accepted we have that $m = \widetilde{\mathsf{rcv}}^p_r\sigma$ where $f_i(\mathsf{sid}) = (\sigma, r, p+1)$. Suppose that $p \geq 1$ and $\mathsf{rcv}^p_r \neq \mathsf{init}$. Then, since $\overline{\widetilde{\mathsf{rcv}}^p_r} = \mathsf{nonces}$, we have that $\overline{m} = \sigma(\mathsf{nonces})$. The converse also holds, that is if $\overline{m}$ is defined then $p \geq 1$ and $\mathsf{rcv}^p_r \neq \mathsf{init}$.

The following property says that in the same role session accepted and sent messages have the same nonces field: If $\mathsf{sid} \in \mathsf{SId}_n$ and $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha_{i'} = \mathbf{send}(\mathsf{sid}, m')$ are two transitions in tr such that m and m' are accepted then m and m' have the same nonces field, provided it is defined for both messages. Indeed the nonces field is given by the substitution in $f_i(\mathsf{sid})$ and $f_{i'}(\mathsf{sid})$ respectively (as we have seen in the previous paragraph). And these substitution are equal on $A_1, \ldots, A_k, N^0_{A_1}, \ldots, N^0_{A_k}$ since they extend the substitution in $f_{i_0}(\mathsf{sid})$ with $i_0 < \min(i, i')$ where i_0 is such that $\mathsf{SId}_{i_0} \setminus \mathsf{SId}_{i_0 - 1} = \{\mathsf{sid}\}$ (that is, α_{i_0} is the **new** transition produced when the session sid was initiated).

Next, we define the relation $\sim$ between role sessions. Intuitively this relation should capture the notion of *protocol session*. That is, two role sessions $(\cdot, r, \cdot)$ and $(\cdot, r', \cdot)$ should be in relation $\sim$ if and only if the two agents playing roles r and r' are communicating in the same protocol session.

We say that two sessions ids $\mathsf{sid}, \mathsf{sid}' \in \mathsf{SId}_n$ are in relation $\sim$ if *there are* two (not necessarily different) transitions in tr labelled by $\alpha = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha' = \mathbf{send}(\mathsf{sid}', m')$ such that m and

[20] Hence $\bar{\cdot}$ is a partial function from terms to terms and $\perp$ means undefined.

[21] The definition of $\widetilde{\mathsf{rcv}}^p_r$ for $p \geq 1$ and the following paragraph provide an explanation for choosing this name. Recall that $\mathsf{nonces} = \langle A_1, \ldots, A_k, N^0_{A_1}, N^0_{A_2}, \ldots, N^0_{A_k} \rangle$ and $\widetilde{\mathsf{rcv}}^p_r = \{\![\mathsf{rcv}^p_r, [\![\mathsf{rcv}^p_r, p', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r'})}]\!\}_{\mathsf{ek}'(A_r)}$.

m' are accepted, $\overline{m} = \overline{m'}$ and $\overline{m} \neq \perp$, that is nonces is instantiated by the same term in the two messages m and m'.

This relation says in fact more about two role sessions: If $\mathsf{sid} \sim \mathsf{sid}'$ then *for any* two transitions in tr labelled by $\alpha = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha' = \mathbf{send}(\mathsf{sid}', m')$ such that m and m' are accepted and $\perp \notin \{\overline{m}, \overline{m'}\}$, we have that $\overline{m} = \overline{m'}$. This is easy to verify using the above stated property (that is, messages which are sent and accepted in the same role session have the same nonces field). Another direct consequence is that if in a session sid the agent executing this session started the second phase (that is he received a valid message m with $\overline{m} \neq \perp$) then $\mathsf{sid} \sim \mathsf{sid}$.

But there may be sessions in which agents are still in the initialisation phase. In these sessions the messages m sent so far have no nonces field and thus the relation $\sim$ doesn't capture them (it is not "defined" on these sessions). However we are not interested in these role sessions and so we do not group them into protocol sessions. But technically we need a partition of all role sessions, hence we simply consider the reflexive closure of $\sim$, denoted $\sim'$. This means that those $\mathsf{sid} \in \mathsf{SId}_n$ for which there is no transition labelled by $\mathbf{send}(\mathsf{sid}, m)$, with m accepted and $\overline{m} \neq \perp$, are only in relation with themselves. The relation $\sim'$ is clearly an equivalence relation. We consider PrtSId to by the quotient set of $\mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ by $\sim'$.

We prove next that the partition PrtSId satisfies the conditions in the definition of honest executions.

Let us look at the first point of Definition 5.11. Consider two arbitrary session ids $\mathsf{sid}, \mathsf{sid}' \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ such that $\mathsf{sid} \sim' \mathsf{sid}'$ and $\mathsf{sid}' \neq \mathsf{sid}$. Let $\mathsf{sid} = (s, r, (a_1, \ldots, a_k))$ and $\mathsf{sid} = (s', r', (a'_1, \ldots, a'_k))$. By the definition of $\sim$ we have that there are two transitions $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ and $\alpha_{i'} = \mathbf{send}(\mathsf{sid}', m')$ such that m and m' have the same nonces field (besides other things). Let $f_i(\mathsf{sid}) = (\sigma, p+1)$ and $f_{i'}(\mathsf{sid}') = (\sigma', p'+1)$. We have that $\sigma(\mathsf{nonces}) = \sigma'(\mathsf{nonces})$. It follows that $A_j\sigma = A_j\sigma'$, that is $a_j = a'_j$ for all $1 \leq j \leq k$. We know that $N^0_{A_r}\sigma = \mathsf{n}^{a_r,0,s}$ and $N^0_{A_{r'}}\sigma' = \mathsf{n}^{a_{r'},0,s'}$. If $r = r'$ then we have in addition that $\mathsf{n}^{a_r,0,s'} = \mathsf{n}^{a_{r'},0,s'}$, thus $s = s'$ which is in contradiction with $\mathsf{sid} \neq \mathsf{sid}'$. Hence $r \neq r'$.

Finally, we prove the second point of Definition 5.11. Let i be the index of the analysed transition $\alpha_i = \mathbf{send}(\mathsf{sid}, m)$ with $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$, m accepted, $m \neq \mathsf{fake}$ and $m = \widetilde{\mathsf{rcv}}^p_r\sigma$, where $\mathsf{sid} = (\cdot, r, \cdot)$ and $f_i(\mathsf{sid}) = (\sigma, p+1)$.

Since tr is an execution trace, $H_{i-1} \vdash \widetilde{\mathsf{rcv}}^p_r\sigma$ holds. Consider a minimal proof associated with this deduction. We have $\widetilde{\mathsf{rcv}}^p_r = \{\![\mathsf{rcv}^p_r, [\![\mathsf{rcv}^p_r, p'', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r''})}]\!\}_{\mathsf{ek}'(A_r)}$ where $(r'', p'') = \mathcal{S}(r, p)$. Since $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ it follows that $A_{r''}\sigma$ is a non-corrupted agent. Hence $H_{i-1} \not\vdash \sigma(\mathsf{sk}'(A_{r''}))$. Thus the message $m_1 = \sigma([\![\mathsf{rcv}^p_r, p'', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r''})})$ was not obtained by a composition rule. Thus, in both cases: m obtained by a composition rule or by a decomposition rule, it follows that m_1 is a subterm of a term t' in H_{i-1}. The term t' was sent at some previous step. Thus there is $i' \leq i$ and $\mathsf{sid}' \in \mathsf{SId}_n$ such that $\alpha_{i'} = \mathbf{send}(\mathsf{sid}', m')$ for some $m' = \widetilde{\mathsf{rcv}}^{p'}_{r'}\sigma'$ and $t' = \widetilde{\mathsf{snt}}^{p'}_{r'}\sigma'$ where $f_{i'}(\mathsf{sid}') = (\sigma', r', p'+1)$. Suppose i' is the smallest such index, that is m_1 is not a subterm of a term of $H_{i'-1}$. We can then have two possibilities.

In the first one, m_1 is a subterm of $\mathsf{snt}^{p'}_{r'}\sigma'$. Since $\mathsf{snt}^{p'}_{r'}$ cannot contain the signature $\mathsf{sk}'(\cdot)$ (the source protocol is constructed over Σ), m_1 is a subterm of $X\sigma'$ where X is a variable of $\mathsf{snt}^{p'}_{r'}$. Hence m_1 is also a subterm of $\mathsf{rcv}^{p'}_{r'}\sigma'$ and moreover a subterm of $m' = \widetilde{\mathsf{rcv}}^{p'}_{r'}\sigma'$. But we have that $H_{i'-1} \vdash m'$. Consider m'_1 be the signed component of m'. Again m'_1 can be obtained only by a decomposition rule. Hence again by the locality lemma, m'_1 is a subterm of a term of $H_{i'-1}$. But m_1 is a subterm of m'_1. We have thus obtained a contradiction (i is not the smallest index such that m_1 is a subterm of a term of H) which means this case doesn't occur.

In the second possibility, $m_1 = [\![\mathsf{snt}_{r'}^{p'}, p', \mathsf{nonces}]\!]_{\mathsf{sk}'(A_{r'})}\sigma'$. It follows that $\sigma(\mathsf{nonces}) = \sigma'(\mathsf{nonces})$ which implies that $\mathsf{sid} \sim \mathsf{sid}'$. We also have $p' = p''$. From $A_{r''}\sigma = A_{r'}\sigma'$ we obtain that $r' = r''$. Let r'_d, p'_d be such that $\mathcal{S}(r'_d, p'_d) = (r', p')$. They exists and are unique by the definition of executable protocols. But since $\mathcal{S}(r, p) = (r'', p'')$ and $(r'', p'') = (r', p')$ it follows that $r'_d = r$ and $p'_d = p$. Finally, since also $\mathsf{snt}_{r'}^{p'}\sigma' = \mathsf{rcv}_r^p\sigma$, we obtain that $m = \widetilde{\mathsf{snt}}_{r'}^{p'}$.

5.5.6.2 Proof of Lemma 5.13

First, note that all terms $t \in H$ are equal to $\widetilde{\mathsf{snt}}_r^p\sigma'$ for some f', sid', r and p with $\mathsf{sid}' = (\cdot, r, \cdot)$, $f'(\mathsf{sid}') = (\sigma', p+1)$ and if $p \geq 1$ and $\mathsf{rcv}_r^p \neq \mathsf{init}$ then these terms are of the form $\{\![m', [\![m', p, \sigma'(\mathsf{nonces})]\!]_{\mathsf{sk}'(a)}]\!\}_{\mathsf{ek}'(b)}$. Second, remark that it is sufficient to prove the desired property for all $t \in H$. The generalisation to deducible messages follows easily. Hence it is sufficient to prove that whenever $X\sigma$ occurs in some $\widetilde{\mathsf{snt}}_r^p\sigma'$ then $\mathsf{rcv}_r^p \neq \mathsf{init}$, $p \geq 1$ and sid' is an honest session id.

Let $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ be a trace of $\widetilde{\Pi}$ and X be a variable of Π. We suppose without loss of generality that $X = N_{A_{\overline{r}}}^j$ for some $\overline{r} \in [k]$ and $j > 0$. Take (SId, f, H) an arbitrary global state of tr and let i be the index of this global state in tr. Consider an honest session id $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ and let $\mathsf{sid} = (s_0, r_0, \cdot)$ and $f(\mathsf{sid}) = (\sigma, \cdot)$.

We prove first that $N_{A_{\overline{r}}}^j\sigma$ is a nonce created in an honest session.

If $\overline{r} = r_0$ then we have that $N_{A_{\overline{r}}}^j\sigma = \mathsf{n}^{a_{\overline{r}}, j, s}$. Suppose $\overline{r} \neq r_0$. This means that $N_{A_{\overline{r}}}^j$ was not initialised in sid by a **new** transition but by a $\alpha_{i_0} = \mathbf{send}(\mathsf{sid}, m)$ transition with $i_0 < i$, m accepted, $m \neq \mathsf{init}$ and $p_0 \geq 1$, where $f_{i_0}(\mathsf{sid}) = (\sigma_0, p_0 + 1)$. We have $N_{A_{\overline{r}}}^j\sigma = N_{A_{\overline{r}}}^j\sigma_0$. Let $(r_1, p_1) = \mathcal{S}(r_0, p_0)$. Since tr is an honest trace (by Lemma 5.12), there are $\mathsf{sid}_1 \in [\mathsf{sid}]$ and $i_1 < i_0$ such that $\mathsf{sid}_1 = (s_1, r_1, \cdot)$ and $f_{i_1}(\mathsf{sid}_1) = (\sigma_1, p_1 + 1)$ for some uid s_1 and substitution σ_1. We have $N_{A_{\overline{r}}}^j\sigma_0 = N_{A_{\overline{r}}}^j\sigma_1$. If $\overline{r} = r_1$ then $N_{A_{\overline{r}}}^j\sigma_1 = \mathsf{n}^{a_{\overline{r}}, j, s_1}$ where $\mathsf{sid}_1 = (s_1, r_1, \cdot)$. Otherwise, continuing in the same way for at most i steps we will certainly find some index l, $0 \leq l < k$ such that $\overline{r} = r_l$ (this is because there are k different roles). Hence anyhow $N_{A_{\overline{r}}}^j\sigma = \mathsf{n}^{a_{\overline{r}}, j, s_l}$. To ease the notation we denote it by n.

If $H_i = H_{i-1}$ then it is sufficient to prove the property for $i-1$. Hence consider that i is such that $H_i \setminus H_{i-1} \neq \emptyset$. It follows that $\alpha_i = \mathbf{send}(\mathsf{sid}', m)$ for some $\mathsf{sid}' \in \mathsf{SId}_i$ (clearly $\alpha_i \neq \mathbf{corrupt}$ since $H_1 \nvdash \mathsf{n}$). Also $m = \widetilde{\mathsf{rcv}}_r^p\sigma'$ where $f_i(\mathsf{sid}') = (\sigma', p+1)$.

We reason by induction on i.

Suppose that i is the *smallest* index such that n occurs in a term of H_i. It follows that $\mathsf{n} \in \mathrm{st}(\widetilde{\mathsf{snt}}_r^p\sigma')$. Then $\mathsf{n} \in \mathrm{st}(Y\sigma')$ where Y is a variable of $\widetilde{\mathsf{snt}}_r^p$.

If Y is not a variable of $\widetilde{\mathsf{rcv}}_r^p$ then from the definition of executable protocols we know that $Y\sigma'$ is a new nonce or key, or an agent identity. Hence $Y = N_{A_r}^{j'}$ or $K_{A_r}^{j'}$ or A' for some j' and A'. That is, $Y\sigma'$ is a constant just like n; thus $Y\sigma' = \mathsf{n}$. Since $\mathsf{n} = \mathsf{n}^{a_r, j, s_l}$ it follows that $A_r\sigma' = a_{\overline{r}}$, $j = j'$ and $\mathsf{sid}' = (s_l, r, \cdot)$. Hence $\mathsf{sid}' = \mathsf{sid}_l$. This means that sid' is an honest session id. Suppose $p < 1$. Then $Y = N_{A_r}^0$ and thus $j = 0$ which is a contradiction. Hence $p \geq 1$ and thus in this case the property is true.

Otherwise, if Y is a variable of $\widetilde{\mathsf{rcv}}_r^p$ then $\mathsf{n} \in \mathrm{st}(\widetilde{\mathsf{rcv}}_r^p\sigma')$, that is $\mathsf{n} \in \mathrm{st}(m)$. Since $H_{i-1} \vdash m$, it follows that $\mathsf{n} \in \mathrm{st}(H_{i-1})$, which is in contradiction with i being the smallest index such that $\mathsf{n} \in \mathrm{st}(H_i)$.

Suppose now that i is *arbitrary*. We have $\mathsf{n} \in \mathrm{st}(H_i \cup \{\widetilde{\mathsf{snt}}_r^p\sigma'\})$. For the occurrence of n in H_{i-1} then the conclusion follows by induction hypothesis. Consider an occurrence of n in $\widetilde{\mathsf{snt}}_r^p\sigma'$. If n occurs in $Y\sigma'$ where Y is not a variable of $\widetilde{\mathsf{rcv}}_r^p$ then, as in the previous paragraph, the conclusion simply follows. Suppose that n occurs in $Y\sigma$ where Y is a variable of $\widetilde{\mathsf{rcv}}_r^p$. Then n occurs in m. Since m is deducible from H_{i-1} and in H_{i-1} all occurrences of n are as required by the induction hypothesis, it follows that the same thing happens in m. That is, $m = m''[\widetilde{\mathsf{snt}}_{r'}^{p'}\sigma'']$ and n occurs in $\widetilde{\mathsf{snt}}_{r'}^{p'}\sigma''$ where $f_i(\mathsf{sid}'') = (\sigma'', p'+1)$ for some $p' \geq 1$ and honest session $\mathsf{sid}'' \in \mathsf{SId}_i$ with $\mathsf{sid}'' = (\cdot, r', \cdot)$. If the occurrence of $\widetilde{\mathsf{snt}}_{r'}^{p'}\sigma''$ in $m = \widetilde{\mathsf{rcv}}_r^p\sigma'$ is in $Y\sigma'$ then the conclusion follows, as this means that $\widetilde{\mathsf{snt}}_{r'}^{p'}\sigma''$ occurs in $\widetilde{\mathsf{snt}}_r^p\sigma'$. Otherwise it must be the case that $m = \widetilde{\mathsf{snt}}_{r'}^{p'}\sigma''$. Then $\sigma'(\mathsf{nonces}) = \sigma''(\mathsf{nonces})$. And since sid'' is an honest session id and $p' \geq 1$ we obtain that also sid' is also an honest session id and $p \geq 1$.

5.5.6.3 Proof of Theorem 5.10

To formally prove the transfer theorem, we need one more lemma, which says that every honest protocol session in the transformed protocol executes exactly like an honest protocol session in the initial protocol without intruder interference. To state this formally we need some auxiliary definitions first.

Since for a session in the transformed protocol there are more actions (corresponding to the initial phase), we define the actions we are interested in.

Let Π be an executable protocol. For an honest trace $\mathsf{tr} = (\mathsf{SId}_0, f_0, H_0) \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_n} (\mathsf{SId}_n, f_n, H_n)$ and a partition $[\mathsf{sid}]$ where $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$ we define $\mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}])$ to be the set of indexes i such that:

- $\alpha_i = \mathbf{new}(r, a_1, \ldots, a_k)$, where $\mathsf{sid} = (\cdot, \cdot, (a_1, \ldots, a_k))$, or
- $\alpha_i = \mathbf{send}(\mathsf{sid}', m)$ and $\mathsf{sid}' \in [\mathsf{sid}]$, m accepted, and $m = \mathsf{rcv}_r^p\sigma$, $p \geq 1$, where $\mathsf{sid}' = (\cdot, r, \cdot)$ and $f_i(\mathsf{sid}') = (\sigma, p+1)$.

Note that the definition of $\mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}])$ does not depend on the representative sid.

Also, we write $\mathsf{Exec}^{\mathsf{h}}(\Pi)$ for the set of honest execution traces of Π.

Lemma 5.14 *Let Π be a protocol and $\widetilde{\Pi}$ the corresponding transformed protocol. Then* $\forall \mathsf{tr} \in \mathsf{Exec}^{\mathsf{h}}(\widetilde{\Pi}), \forall \mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$*, there are* $\mathsf{tr}_0 \in \mathsf{Exec}^{p,1}(\Pi)$, $\mathsf{sid}_0 \in \mathsf{SId}(\mathsf{tr}_0)$ *and bijections* $\mathcal{I} \colon \mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}]) \to \mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_0])$, $g : \mathsf{Ag}(\mathsf{sid}) \to \mathsf{Ag}(\mathsf{sid}_0)$ *and* $\varphi : [\mathsf{sid}] \to [\mathsf{sid}_0]$ *such that* $\forall \mathsf{sid}' \in [\mathsf{sid}]$*, the same role plays in* sid' *and* $\varphi(\mathsf{sid}')$*, and* $\forall i \in \mathsf{Ix}(\mathsf{tr})$, $f^0_{\mathcal{I}(i)}(\varphi(\mathsf{sid}')) = (\sigma_0, p)$ *with* $\sigma = \sigma_0 \circ g$ *where* $f_i(\mathsf{sid}') = (\sigma, p)$, $\mathsf{tr} = (f_i, \cdot, \cdot)_i$ *and* $\mathsf{tr}_0 = (f^0_j, \cdot, \cdot)_j$*. Moreover, for these* tr_0, sid_0 *and bijections the converse also holds, that is,* $\forall \mathsf{sid}'_0 \in \mathsf{SId}(\mathsf{tr}_0), \forall i_0 \in \mathsf{Ix}(\mathsf{tr}_0)$, $f_{\mathcal{I}^{-1}(i_0)}(\varphi^{-1}(\mathsf{sid}'_0)) = (\sigma, p)$ *with* $\sigma = \sigma_0 \circ g$ *where* $f^0_{i_0}(\mathsf{sid}'_0) = (\sigma_0, p)$.

The proof of this lemma consists of a simple rewriting of the definition of honest traces into the definition of honest, single session traces.

We now proceed with the proof of the theorem. Consider an arbitrary closed formula $\phi \in \mathcal{L}'$ such that $\Pi \models^{p,1} \phi$.

Let $\mathsf{tr} \in \mathsf{Exec}(\widetilde{\Pi}) = (\mathsf{SId}_\iota, f_\iota, H_\iota)_{1 \leq \iota \leq n}$. From Lemma 5.12 we know that $\mathsf{tr} \in \mathsf{Exec}^{\mathsf{h}}(\widetilde{\Pi})$. Also let $\sigma \in \mathcal{LS}_{r,p}(\mathsf{tr})$ such that $\mathsf{NC}(\sigma(A_l))$ holds for all $1 \leq l \leq k$. Hence there are an index ι with $1 \leq \iota \leq n$ and a session id $\mathsf{sid} \in \mathsf{SId}(\mathsf{tr})$ such that $\mathsf{sid} = (\cdot, r, \cdot)$ and $f_\iota(\mathsf{sid}) = (\sigma, p)$. Moreover, $\mathsf{sid} \in \mathsf{SId}^{\mathsf{h}}(\mathsf{tr})$. We can suppose that $\iota \in \mathsf{Ix}(\mathsf{tr}, \mathsf{sid})$ because otherwise it would be easy to find another index which has this property.

Applying Lemma 5.14 we obtain that there are $\mathsf{tr}_0 \in \mathsf{Exec}^{p,1}(\Pi)$, $\mathsf{sid}_0 \in \mathsf{SId}(\mathsf{tr})$ and bijections $\mathcal{I} : \mathsf{Ix}(\mathsf{tr}, [\mathsf{sid}]) \to \mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_0])$, $g : \mathsf{Ag}(\mathsf{sid}) \to \mathsf{Ag}(\mathsf{sid}_0)$ and $\varphi : [\mathsf{sid}] \to [\mathsf{sid}_0]$ satisfying certain properties. In particular, if we let $\mathsf{sid}_1 = \varphi(\mathsf{sid})$ and $\iota_0 = \mathcal{I}(\iota)$ then we have $f^0_{\iota_0}(\mathsf{sid}_1) = (\sigma_0, p)$ with $\sigma = \sigma_0 \circ g$.

Also, since tr_0 is an honest, single session trace by its definition, we have that sid_1 is an honest session id. Then $\mathsf{NC}(\sigma_0(A_l))$ is true in tr_0 for all $1 \leq l \leq k$. From the hypothesis we know that $[\![\phi, \mathsf{tr}_0]\!] = 1$. Hence, since the left hand side of the implication holds for tr_0, it follows that also the right hand side holds for tr_0 and $\sigma_0 \in \mathcal{LS}_{r,p}(\mathsf{tr}_0)$, that is for ι_0 and sid_1.

Fix an arbitrary i. What we have to prove depends on the form of the subformula in the right hand side of the implication.

Consider first that $\mathcal{Q}_i = \exists$. Since $[\![\phi, \mathsf{tr}_0]\!] = 1$, there exist ι'_0 in $\mathsf{Ix}(\mathsf{tr}_0, [\mathsf{sid}_1])$ and $\mathsf{sid}'_0 \in \mathsf{SId}(\mathsf{tr}_0)$ such that the formulas $\tau^i_j(u^i_j, v^i_j)[{}^{\sigma_0}/_{\varsigma}][{}^{\sigma'_0}/_{\varsigma'}]$ hold for all $j \in J_i$, where $\mathsf{sid}'_0 = (\cdot, r_i, \cdot)$ and $f^0_{\iota'_0}(\mathsf{sid}'_0) = (\sigma'_0, p_i)$. Let $\iota' = \mathcal{I}^{-1}(\iota'_0)$ and $\mathsf{sid}' = \varphi^{-1}(\mathsf{sid}'_0)$. Again by Lemma 5.14 we have that $f_{\iota'}(\mathsf{sid}') = (\sigma', p_i)$ with $\sigma' = \sigma'_0 \circ g$. Since σ, σ' are equal with σ_0, σ'_0 respectively, modulo the same bijective renaming g of agent identities, then it follows easily that $\tau^i_j(u^i_j, v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma'}/_{\varsigma_i}]$ are true for all $j \in J_i$. Hence the formula $\exists \varsigma_i \in \mathcal{LS}_{r_i,p_i} \bigwedge_{j \in J_i} \tau^i_j(u^i_j, v^i_j)$ is true.

Consider now that $\mathcal{Q}_i = \exists!$. The existence of ι' and sid' is assured as in the previous paragraph. Let $\varsigma(X)$ with X a nonce (or key) variable be a subterm in u^i_j or v^i_j for some $j \in J_i$ with $\tau^i_j \in \{=\}$.

Concerning uniqueness, assume there exist $\mathsf{sid}'' \in \mathsf{SId}(\mathsf{tr})$ and $\iota'' \in \mathsf{Ix}(\mathsf{tr})$ such that, in particular $(u^i_j = v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma''}/_{\varsigma_i}]$, where $\mathsf{sid}'' = (\cdot, r_i, \cdot)$ and $f_{\iota''}(\mathsf{sid}'') = (\sigma'', p_i)$. Consider an occurrence of $\varsigma(X)$ say in u^i_j, at position q. There is an occurrence q' in v^i_j with $q' \leq q$ such that $(v^i_j)|_{q'} = \varsigma(Y)$ or $(v^i_j)|_{q'} = \varsigma_i(Y)$ where Y is a variable. Since we have uniqueness in the passive, single session case then $(v^i_j)|_{q'} = \varsigma_i(Y)$. Hence $X\sigma$ occurs in both $Y\sigma'$ and $Y\sigma''$.

If Y was received in session sid'' then there is an action $\alpha_{\iota''_1} = \mathbf{send}(\mathsf{sid}'', m)$ such that $m = \widetilde{\mathsf{rcv}}^{p'}_{r_i}\theta'$, $f_{\iota''_1}(\mathsf{sid}'') = (\theta', p'+1)$, $f_{\iota''_1-1}(\mathsf{sid}'') = (\theta, p')$ and θ was not defined on Y. We also have σ'' extends θ hence in particular $Y\sigma'' = Y\theta$. If Y was created (i.e. was initialised by a **new** action) in sid'' then it was also sent within some message $m = \widetilde{\mathsf{snt}}^{p'}_{r_i}\theta$, again with σ'' extending θ. In both cases, since m is deducible from the intruder's knowledge and $X\sigma$ occurs in m we can apply now Lemma 5.13 to obtain that sid'' is an honest session id and $\sigma''(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. If Y was also received in session sid' then we can prove similarly that $\sigma'(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. Intuitively, different role sessions can't be played by the same role (i.e. r_i) in the same protocol session hence $\mathsf{sid}' = \mathsf{sid}''$. Formally, this is obtained from the equality $N^0_{A_{r_i}}\sigma' = N^0_{A_{r_i}}\sigma''$ taking into account that $N^0_{A_{r_i}}$ was initialised in both sessions.

Finally consider that $\mathcal{Q}_i = \forall$.

Suppose that there are ι' and sid' such that $\tau^i_j(u^i_j, v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma'}/_{\varsigma_i}]$ does not hold for some $j \in J_i$ where $\mathsf{sid}' = (\cdot, r_i, \cdot)$ and $f_{\iota'}(\mathsf{sid}') = (\sigma', p_i)$. That is $(u^i_j = v^i_j)[{}^{\sigma}/_{\varsigma}][{}^{\sigma'}/_{\varsigma_i}]$. Let $\varsigma(X)$ with X a nonce (or key) variable be a subterm in u^i_j (the case v^i_j is symmetric). Then, as before $X\sigma$ occurs in $Y\sigma''$ where $\sigma'' = \sigma$ or $\sigma'' = \sigma'$. If $\sigma'' = \sigma'$ then again using Lemma 5.13 it follows that sid' is an honest session and $\sigma'(\mathsf{nonces}) = \sigma(\mathsf{nonces})$. Hence $\mathsf{sid}' \in [\mathsf{sid}]$. Let $\iota'_0 = \mathcal{I}(\iota')$ and $\mathsf{sid}'_0 = \varphi(\mathsf{sid}'_0)$. We have (from Lemma 5.14 again) that $\sigma' = \sigma'_0 \circ g$. Hence $(u^i_j = v^i_j)[{}^{\sigma_0}/_{\varsigma}][{}^{\sigma'_0}/_{\varsigma_i}]$ which is a contradiction with the hypothesis for tr_0, ι_0 and σ_0. Hence the supposition we made is false.

5.6 Conclusions

We have presented a general transformation for security protocols that essentially prevents an active adversary to interfere with the executions of the protocol that involves only honest parties. An important consequence of our transformation is that it enables a transference theorem of a non-trivial class of security properties from a setting where no adversary is present to a setting where a fully active adversary may tamper with the protocol execution. The security properties that are transferred include secrecy and various formulations of authentication.

Finally, our transformation makes quite heavy use of expensive cryptographic primitives. It is thus important to look for simpler transformations, and several possibilities can be explored. We could exchange and authenticate encryption keys in the preliminary phase by using the existing public key infrastructure, instead of using a new such infrastructure. We could also use other primitives like signcryption (a public key cryptosystem for both signing and encrypting), symmetric encryptions and macs, or use hybrid encryption (combine public and symmetric encryption).

Conclusions and perspectives

In this thesis we have contributed to the analysis of security protocols in symbolic models by investigating less explored cryptographic primitives, security properties, and approaches to protocol analysis.

Concretely, the work done in this thesis is summarised below:

- We have formulated the constraint system approach [MS01] for arbitrary trace security properties and we have proved that its complexity is NP-time, as long as the security property can be decided in polynomial time on simpler constraint systems (i.e. on solved forms). As a consequence, we obtain an alternative proof of the complexity (i.e. NP-completeness) result [RT01] for secrecy for a bounded number of sessions, in the context of constraint systems.

- We have applied the mentioned generic approach to the problem of detecting key cycles and proved that this problem is NP-complete for a bounded number of sessions. As another application of this approach, we have also showed that secrecy remains NP-complete for protocols which use timestamps.

- We have provided a resolution strategy for deciding a new class of Horn clauses modeling protocols which use CBC encryption or blind signatures. We have applied this strategy to the Needham-Schroeder symmetric key protocol which has a flaw when implemented with CBC encryption. We have fixed the protocol and automatically proved the correctness of the fixed version of the protocol.

- We have related the two standard secrecy notions, "simple" (reachability-based) secrecy and "strong" (equivalence-based) secrecy, by giving sufficient syntactic conditions on the protocols for simple secrecy to imply strong secrecy. In this way, for (the class of) protocols satisfying these conditions, we are able to transfer the existing results obtained for simple secrecy to strong secrecy. As examples, we proved that the Yahalom, Otway-Rees, and Wide-Mouthed-Frog protocols preserve the strong secrecy of exchanged keys for an unbounded number of sessions.

- We have presented a transformation that maps a protocol secure in an extremely weak sense (essentially in a model where no adversary is present) into a protocol that is secure against a fully active adversary which interacts with an unbounded number of protocol sessions. The transformation preserves a large class of trace security properties containing secrecy and authentication.

Perspectives

Models for security protocols We have basically used two kinds of models for specifying security protocols in this thesis: one using pattern-matching and one using explicit destructors. It would be interesting to know precisely the differences between them, both at the modeling level and at the level of security guarantees. That is, is one of them able to express more protocols, or more faithfully some protocols? And are there attacks that can be captured within one model and cannot be captured within the other? J. Millen [Mil03], and Ch. Lynch and C. Meadows [LM05] have performed such a comparison, but only in concrete settings (i.e. for symmetric and asymmetric encryption respectively), while we would like to work in a general setting (e.g. with arbitrary primitives exhibiting algebraic properties). We believe that we have already set up in Chapter 1 a part of the formalism necessary to perform such a comparison.

Constraint systems and key cycles For a bounded number of sessions, we have treated arbitrary trace properties by expressing them as predicates on lists of messages. It would be nicer to express properties by formulas in some logic, as was done for example by R. Corin *et al* [CSE05] using a variant of LTL. However, to decide such security properties, they used the Millen-Shmatikov procedure as a black box, while we could also obtain the complexity of checking them.

We have handled several notions of key cycles. However, still other variants of key cycles (or similar conditions) may already exist or appear in the future. It would then be nice to have a formalism which would allow to verify such properties in a modular manner.

Also, our approach is valid for a bounded number of sessions only. Secrecy is undecidable in general [DLM04] for an unbounded number of sessions. Such an undecidability result could be easily adapted to the problem of detecting key cycles. Several decidable fragments have been designed [RS03, CLC03a, BP03b, VSS05] for secrecy and an unbounded number of sessions. We plan to investigate how such fragments could be used to decide key cycles. An approach could be to encode Laud's deduction system [Lau02] (for detecting key cycles in the passive case) into Horn clauses, and then to reuse or extend an existing fragment of Horn clauses to decide the satisfiability of the resulting set of clauses.

From a practical point of view, as the CL-AtSe back-end [Tur06] of the Avispa tool [ABB$^+$05] basically shares the same underlying ideas as the constraint system approach, we hope that CL-AtSe can be relatively easily extended in order to handle key cycles and timestamps.

Transformation from insecure to secure protocols In Chapter 3, we have applied our resolution strategy for debugging of a protocol under a more realistic threat model than the one usually considered. We have transformed this protocol so that it falls into the scope of our Horn class. This transformation preserves the attacks and therefore the correctness of the target protocol ensures the correctness of the initial one. The transformation is interesting in itself. We would like to further investigate this type of transformations and to characterise the protocols to which they can be safely applied.

From simple secrecy to strong secrecy We plan to further investigate the active case of our transfer result by trying to relax our conditions. There are several possible directions. Firstly, we may consider specific classes of protocols by restricting the syntax (for instance considering ping-pong protocols such as in [AC02, HS05]) to see whether it is possible to refine our results in this setting. Secondly, we may relax the requirement that processes cannot test over the secret by requiring instead that the two branches of the test are indistinguishable. This is the case for example when a test is followed in each branch by other tests that will never succeed when the first one is really applied to a secret data. This would require to consider more complex over-approximations of the set of sent messages. In particular, in the definition of the set $\mathcal{E}(P)$, we would have to consider trees instead of just paths potentially leading to the secret.

Transformation to obtain secure protocols Our transfer result was established for protocols using standard Dolev-Yao primitives. We believe that we could relatively easily extend the result such that to allow arbitrary primitives (with their properties) in the initial protocol. We also plan to investigate compositionality issues related to the transformation. For example, is the transformed protocol still secure when used in parallel with other (non-)transformed protocols?

One interesting avenue for future research it to obtain more general transference theorems between the properties of the original protocol and those of the transformed protocol. It would be also interesting to investigate the modular development approach implied by our results. In particular, it would be interesting to design a language for building "naive" specification which can then be compiled into secure protocols using our transformation.

Finally, from an efficiency perspective, it is important to look for simpler transformations that make lighter use of cryptographic primitives, perhaps at the expense of ensuring weaker security guarantees for the resulting protocol. We note that the Katz and Yung compiler [KY03] is one example of such a transformation which deserves further investigation.

Bibliography

[AB02] Martín Abadi and Bruno Blanchet. Analyzing Security Protocols with Secrecy Types and Logic Programs. In *29th Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages (POPL 2002)*, pages 33–44, Portland, Oregon, January 2002. ACM Press.

[Aba00] M. Abadi. Security protocols and their properties. In *20th Int. Summer School, Marktoberdorf, Germany*, pages 39–60. IOS Press, 2000.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.C. 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 applications. In *Proc. of Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, 2005.

[ABHS05] P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 374–396, 2005.

[AC02] R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. CONCUR 02. Springer-Verlag, 2002.*, 2002.

[AC04] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 46–58. Springer, 2004.

[AC05] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005)*, pages 62–76. IEEE Computer Society, 2005.

[ACD07] Mathilde Arnaud, Véronique Cortier, and Stéphanie Delaune. Combining algorithms for deciding knowledge in security protocols. In Franck 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, September 2007. Springer.

[Adã06] P. Adão. *Formal Methods for the Analysis of Security Protocols.* PhD thesis, IST, Universidade Técnica de Lisboa, June 2006.

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

[AFG02] Martìn Abadi, Cédric Fournet, and Georges Gonthier. Secure implementation of channel abstractions. *Inf. Comput.*, 174(1):37–83, 2002.

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

[AG98] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, 1998.

[AL00] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *12th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 380–394, 2000.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS'00)*, volume 1872 of *LNCS*, pages 3–22, August 2000.

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

[BAF05] B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 331–340. IEEE Computer Society Press, 2005.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[Ban05] Gergei Bana. Soundness and completeness of formal logics of symmetric encryption. Cryptology ePrint Archive, Report 2005/101, 2005. `http://eprint.iacr.org/`.

[Bau05] Mathieu 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, November 2005.

[Bau07] Mathieu Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires.* Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, January 2007.

[BBN04] J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulations in the spi calculus. In *15th Conf on Concurrency Theory (CONCUR'04)*, volume 3170 of *LNCS*, pages 161–176. Springer, 2004.

[BC06] Vincent Bernat and Hubert Comon-Lundh. Normal proofs in intruder theories. In Mitsu Okada and Ichiro Satoh, editors, *Proceedings of the 11th Asian Computing Science Conference (ASIAN'06)*, Lecture Notes in Computer Science, Tokyo, Japan, December 2006. Springer. To appear.

[BCD07] Sergiu Bursuc, Hubert Comon-Lundh, and Stéphanie Delaune. Associative-commutative deducibility constraints. In Wolfgang Thomas and Pascal 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, February 2007. Springer.

[BCJ+06] Michael Backes, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, and Joe-Kai Tsay. Cryptographically sound security proofs for basic and public-key kerberos. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, pages 362–383. Springer, 2006.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428, New York, NY, USA, 1998. ACM Press.

[BCR03] M. Bugliesi, A. Ceccato, and S. Rossi. Context-sensitive equivalences for non-interference based protocol analysis. In *Fundamentals of Computation Theory, 14th International Symposium*, volume 2751 of *Lecture Notes in Computer Science*, pages 364–375. Springer, 2003.

[BEL04] L. Bozga, C. Ene, and Y. Lakhnech. A symbolic decision procedure for cryptographic protocols with time stamps. In *Proc. 15th International Conference on Concurrency Theory (CONCUR'04)*, LNCS, pages 177–192, London, England, 2004. Springer-Verlag.

[BFG04] K. Bhargavan, C. Fournet, and A. D. Gordon. A semantics for web services authentication. In *31st ACM Symposium on Principles of Programming Languages (POPL'04)*, pages 198–209. ACM, January 2004.

[BFGT06] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Stephen Tse. Verified interoperable implementations of security protocols. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006)*, pages 139–152. IEEE Computer Society, 2006.

[BG01] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 19–99. Elsevier and MIT Press, 2001.

[BL06] Michael Backes and Peeter Laud. Computationally sound secrecy proofs by mechanized flow analysis. In *Proceedings of 13th ACM Conference on Computer and Communications Security (CCS)*, pages 370–379, November 2006.

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Comp. Soc. Press, 2001.

[Bla02] Bruno Blanchet. From Secrecy to Authenticity in Security Protocols. In Manuel Hermenegildo and Germán Puebla, editors, *9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *Lecture Notes on Computer Science*, pages 342–359, Madrid, Spain, September 2002. Springer Verlag.

[Bla04] B. Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy (SP'04)*, pages 86–100, Oakland, California, May 2004.

[Bla05] Bruno Blanchet. Security Protocols: From Linear to Classical Logic by Abstract Interpretation. *Information Processing Letters*, 95(5):473–479, September 2005.

[Bla07] Bruno Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 97–111, Venice, Italy, July 2007. IEEE.

[BLP03] L. Bozga, Y. Lakhnech, and M. Périn. HERMES: An automatic tool for verification of secrecy in security protocols. In *15th Int. Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 219–222. Springer, 2003.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[BNP99] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. In *Logic in Computer Science*, pages 157–166, 1999.

[BP03a] Michael Backes and Birgit Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In *Proceedings of the 23rd Conference on the Foundations of Software Technology and Theoretical Computer Science (FST TCS 2003)*, volume 2914 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.

[BP03b] B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In Andrew Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, Warsaw, Poland, April 2003. Springer Verlag.

[BP04] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218, 2004.

[BPS07] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks – BRSIM/UC-soundness of symbolic encryption with key cycles. In *Proc. of 20th IEEE Computer Security Foundation Symposium (CSF)*, June 2007. Preprint on IACR ePrint 2005/421.

[BR93] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93, 13th Annual International Cryptology Conference*, volume 773 of *LNCS*, pages 232–249, 1993.

[CC05] Hubert Comon and Véronique Cortier. Tree automata with one memory set constraints and cryptographic protocols. *Theor. Comput. Sci.*, 331(1):143–214, 2005.

[CD05] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In Jürgen 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, April 2005. Springer.

[CDE05] Ricardo Corin, Jeroen Doumen, and Sandro Etalle. Analysing password protocol security against off-line dictionary attacks. *Electr. Notes Theor. Comput. Sci.*, 121:47–63, 2005.

[CDF+07] Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, Karthikeyan Bhargavan, and James J. Leifer. Secure implementations for typed session abstractions. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 170–186, Venice, Italy, July 2007. IEEE Computer Society.

[CDL+00] Iliano Cervesato, Nancy Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. Relating Strands and Multiset Rewriting for Security Protocol Analysis. In *13th Computer Security Foundations Workshop — CSFW-13*, pages 35–51, Cambridge, UK, 3–5 July 2000. IEEE Computer Society Press.

[CDL06] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[CE02] Ricardo Corin and Sandro Etalle. An improved constraint-based system for the verification of security protocols. In M. Hermenegildo and G. Puebla, editors, *Proceedings of the 9th International Symposium on Static Analysis (SAS'02)*, volume 2477 of *Lecture Notes in Computer Science*, pages 326–341. Springer, September 2002.

[CHW06] Véronique Cortier, Heinrich Hördegen, and Bogdan Warinschi. Explicit Randomness is not Necessary when Modeling Probabilistic Encryption. In *Workshop on Information and Computer Security (ICS 2006)*, Timisoara, Romania, September 2006.

[CJ97] J. Clark and J. Jacob. A survey of authentication protocol literature. Available at `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.

[CJM00] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with Brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4):443–487, 2000.

[CJT+06] Iliano Cervesato, Aaron D. Jaggard, Joe-Kai Tsay, Andre Scedrov, and Christopher Walstad. Breaking and Fixing Public-Key Kerberos. In Mitsu Okada and Ichiro Satoh, editors, *Eleventh Annual Asian Computing Science Conference — ASIAN'06*, pages 164–178, Tokyo, Japan, 6–8 December 2006.

[CKRT03] Y. Chevalier, R. Kuesters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proc. of the Logic In Computer Science Conference LICS'03*, June 2003.

[CL04] H. Comon-Lundh. Résolution de contraintes et recherche d'attaques pour un nombre borné de sessions. Available at `http://www.lsv.ens-cachan.fr/~comon/CRYPTO/bounded.ps`, 2004.

[CLC03a] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 148–164, Valencia (Spain), June 2003. Springer-Verlag.

[CLC03b] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proc. of the 12th European Symposium On Programming (ESOP'03)*, volume 2618 of *Lecture Notes in Computer Science*, pages 99–113, Warsaw (Poland), April 2003. Springer-Verlag.

[CLR07] Yannick Chevalier, Denis Lugiez, and Michaël Rusinowitch. Towards an automatic analysis of web service security. In *6th International Symposium on Frontiers of Combining Systems, FroCoS 2007*, volume 4720 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2007.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 271–280, 2003.

[Cor03] Véronique Cortier. *Vérification automatique des protocoles cryptographiques*. PhD thesis, École Normale Supérieure de Cachan, Cachan (France), March 2003.

[Cor06] R. Corin. *Analysis Models for Security Protocols*. PhD thesis, University of Twente, The Netherlands, Twente (Netherlands), 2006.

[CR06] Yannick Chevalier and Michaël Rusinowitch. Hierarchical combination of intruder theories. In *17th International Conference on Term Rewriting and Applications, RTA 2006*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2006.

[CRZ05] Véronique Cortier, Michaël Rusinowitch, and Eugen Zălinescu. A Resolution Strategy for Verifying Cryptographic Protocols with CBC Encryption and Blind Signatures. In Pedro Barahona and Amy P. Felty, editors, *Proceedings of the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'05)*, pages 12–22, Lisboa, Portugal, July 2005. ACM Press.

[CRZ06] Véronique Cortier, Michaël Rusinowitch, and Eugen Zălinescu. Relating two standard notions of secrecy. In Zoltan Esik, editor, *Proceedings of the 20th International Conference on Computer Science Logic (CSL'06)*, volume 4207 of *Lecture Notes in Computer Science*, pages 303–318, Szeged, Hungary, September 2006. Springer.

[CSE05] R. J. Corin, A. Saptawijaya, and S. Etalle. Ps-ltl for constraint-based security protocol analysis. In M. Gabbrielli and G. Gupta, editors, *Logic Programming, 21st International Conference, ICLP 2005, Sitges, Spain*, volume 3668 of *Electronic Notes in Theoretical Computer Science*, pages 439–440. Springer Verlag, October 2005.

[CW05] V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 157–171. Springer, April 2005.

[CWZ07] Véronique Cortier, Bogdan Warinschi, and Eugen Zălinescu. Synthesizing secure protocols. In Joachim Biskup and Javier Lopez, editors, *Proceedings of the 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 406–421, Dresden, Germany, September 2007. Springer.

[CZ06] Véronique Cortier and Eugen Zălinescu. Deciding key cycles for security protocols. In Miki Hermann and Andrei Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 317 – 331, Phnom Penh, Cambodia, November 2006. Springer.

[DDMP05] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *J. Comput. Secur.*, 13(3):423–482, 2005.

[DEK82] Danny Dolev, Shimon Even, and Richard M. Karp. On the security of ping-pong protocols. *Information and Control*, 55(1-3):57–68, 1982.

[Del06] Stéphanie Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques*. PhD thesis, École Normale Supérieure de Cachan, Cachan (France), June 2006.

[DJ04] Stéphanie Delaune and Florent Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, Washington, D.C., USA, October 2004. ACM Press.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, Venice, Italy, July 2006. IEEE Computer Society Press.

[DLL07] Stéphanie Delaune, Hai Lin, and Christopher Lynch. Protocol verification via rigid/flexible resolution. In Nachum Dershowitz and Andrei 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, October 2007. Springer.

[DLLT06] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis in presence of a homomorphism operator and *exclusive or*. In Michele Buglesi, Bart Preneel, Vladimiro Sassone, and Ingo 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.

[DLLT07] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 2007. To appear.

[DLM04] N. Durgin, P. Lincoln, and J. 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 *Workshop on Formal Methods and Security Protocols*, Trento, Italia, 1999.

[DNL99] B. Donovan, P. Norris, and G. Lowe. Analyzing a library of security protocols using Casper and FDR, July 1999.

[DSV00] L. Durante, R. Sisto, and A. Valenzano. A state-exploration technique for spi-calculus testing equivalence verification. In *Formal Techniques for Distributed System Development (FORTE/PSTV 2000)*, volume 183 of *IFIP Conference Proceedings*, pages 155–170. Kluwer, 2000.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12):198–208, 1983.

[EG83] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 34–39, 1983.

[EHHN99] A. Elkjær, M. Höhle, H. Hüttel, and K. Nielsen. Towards automatic bisimilarity checking in the spi calculus. *Combinatorics, Computation, and Logic: Proceedings of DMTCS'99 and CATS'99*, 21(3):175–189, 1999.

[FB01] Wayne Snyder Franz Baader. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

[FGM00] R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Automata, Languages and Programming*, pages 354–372, 2000.

[FLHT01] Christian Fermüller, Alexander Leitsch, Ullrich Hustadt, and Tanel Tammet. Resolution decision procedures. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 1791–1849. Elsevier and MIT Press, 2001.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology - AUSACRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[Frö07] Sibylle Fröschle. The insecurity problem: tackling unbounded data. In *IEEE Computer Security Foundations Symposium 2007*. IEEE Computer Society, 2007.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18. Springer-Verlag, 1985.

[GK00] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification. In *Conference on Automated Deduction*, pages 271–290, 2000.

[GM82] Joseph A. Goguen and José Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[GM92] Joseph A. Goguen and José Meseguer. Order-sorted algebra i: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theor. Comput. Sci.*, 105(2):217–273, 1992.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[Gou00] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification (extended abstract). In José D. P. Rolim, editor, *Proceedings of the Workshops of the 15th International Parallel and Distributed Processing Symposium*, volume 1800 of *Lecture Notes in Computer Science*, pages 977–984, Cancun, Mexico, May 2000. Springer.

[GP05] Jean Goubault-Larrecq and Fabrice Parrennes. Cryptographic protocol analysis on real C code. In Radhia Cousot, editor, *Proceedings of the 6th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *Lecture Notes in Computer Science*, pages 363–379, Paris, France, January 2005. Springer.

[GS95] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 44–55, 1995.

[HS05] H. Hüttel and J. Srba. Recursion versus replication in simple cryptographic protocols. In *31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 3381 of *LNCS*, pages 178–187, 2005.

[Hui99] A. Huima. Efficient infinite-state analysis of security protocols, 1999.

[Hüt02] H. Hüttel. Deciding framed bisimilarity. In *INFINITY'02*, August 2002.

[Jan06] Romain Janvier. *Lien entre modèles symboliques et computationnels pour le protocoles cryptographiques utilisant des hachage.* PhD thesis, Université Joseph Fourier, Grenoble (France), September 2006.

[JLM05] R. Janvier, Y. Lakhnech, and L. Mazare. (De)Compositions of Cryptographic Schemes and their Applications to Protocols. Cryptology ePrint Archive, Report 2005/020, 2005.

[JRV00] Florent Jacquemard, Michaël Rusinowitch, and Laurent Vigneron. Compiling and verifying security protocols. In *Logic Programming and Automated Reasoning*, pages 131–160, 2000.

[KKT07] D. Kähler, R. Küsters, and T. Truderung. Infinite State AMC-Model Checking for Cryptographic Protocols. In *Proceedings of the Twenty-Second Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 181–190. IEEE, Computer Society Press, 2007.

[KKW05] D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. In V. Diekert and B. Durand, editors, *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, number 3404 in Lecture Notes in Computer Science, pages 158–169. Springer-Verlag, 2005.

[KKW07] K.O. Kürtz, R. Küsters, and Th. Wilke. Selecting theories and nonce generation for recursive protocols. In *FMSE*, 2007. To appear.

[KR05] S. Kremer and M. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi-Calculus. In Mooly Sagiv, editor, *Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., April 2005. Springer-Verlag.

[KY03] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proceedings of Crypto'03*, pages 110–125. Springer-Verlag, 2003.

[Laf06] Pascal Lafourcade. *Vérification des protocoles cryptographiques en présence de théories équationnelles.* Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, September 2006. 209 pages.

[Lau02] P. Laud. Encryption cycles and two views of cryptography. In *Nordic Workshop on Secure IT Systems (NORDSEC'02)*, 2002.

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for *AC*-like equational theories with homomorphisms. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322, Nara (Japan), April 2005. Springer-Verlag.

[LM05] Christopher Lynch and Catherine Meadows. On the relative soundness of the free algebra model for public key encryption. *Electr. Notes Theor. Comput. Sci.*, 125(1):43–54, 2005.

[LMH07] Juan C. López, Raúl Monroy, and Dieter Hutter. On the automated correction of security protocols susceptible to a replay attack. In *Proceedings of the 11th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 594–609. Springer, 2007.

[LMMS98] Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *Lecture Notes on Computer Science*, pages 147–166. Springer-Verlag, 1996.

[Low97] G. Lowe. A hierarchy of authentication specifications. In *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97)*, pages 31–44. IEEE Computer Society Press, 1997.

[Low99] Gavin Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.

[Maz06] Laurent Mazaré. *Computational Soundness of Symbolic Models for Cryptographic Protocols.* Thèse de doctorat, Institut National Polytechnique de Grenoble, France, October 2006.

[McA93] David A. McAllester. Automatic recognition of tractability in inference relations. *Journal of the ACM*, 40(2):284–303, 1993.

[Mea96] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[Mil03] Jonathan K. Millen. On the freedom of decryption. *Inf. Process. Lett.*, 86(6):329–333, 2003.

[MMS97] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Mur-phi. In *IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[Mon99] David Monniaux. Abstracting cryptographic protocols with tree automata. In *Sixth International Static Analysis Symposium (SAS'99)*, number 1694 in Lecture Notes in Computer Science, pages 149–163. Springer Verlag, 1999.

[MS01] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communication Security*, pages 166–175. ACM SIGSAC, November 2001.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[MW04a] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. Theory of Cryptography Conference (TCC'04)*, pages 133–151, 2004.

[MW04b] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.

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

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[Pau01] L. C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *Journal of Computer Security*, 9(3):197–216, 2001.

[PQ00] O. Pereira and J.-J. Quisquater. On the perfect encryption assumption. In *Proc. of the 1st Workshop on Issues in the Theory of Security (WITS'00)*, pages 42–45, Geneva (Switzerland), 2000.

[PS00] Adrian Perrig and Dawn Song. Looking for diamonds in the desert — extending automatic protocol generation to three-party authentication and key agreement protocols. In *Computer Security Foundations Workshop (CSFW '00)*, 2000.

[PSW00] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Cryptographic security of reactive systems. *Electr. Notes Theor. Comput. Sci.*, 32, 2000.

[RS03] R. Ramanujam and S.P.Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, Mumbai, 2003.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, April 2003.

[SBP01] Dawn Xiaodong Song, Sergey Berezin, and Adrian Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[Sch93] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[SP03] E. Sumii and B. Pierce. Logical relations for encryption. *Journal of Computer Security*, 11(4):521–554, 2003.

[Spo] Security protocols open repository. `http://www.lsv.ens-cachan.fr/spore`.

[SV05] H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *Proc. of 11th International Conference on Logic for Programming and Automated Reasoning (LPAR'04)*, volume 3452 of *Lecture Notes in Computer Science*, pages 79–94, Montevideo (Uruguay), 2005. Springer-Verlag.

[SV06] Helmut Seidl and Kumar Neeraj Verma. Cryptographic protocol verification using tractable classes of Horn clauses. In *Program Analysis and Compilation, Theory and Practice, Essays Dedicated to Reinhard Wilhelm on the Occasion of His 60th Birthday*, volume 4444 of *Lecture Notes in Computer Science*, pages 97–119. Springer, 2006.

[Tur06] Mathieu Turuani. The CL-Atse Protocol Analyser. In *Term Rewriting and Applications - Proc. of RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286, Seattle, WA, USA, 2006.

[VIS96] D. Volpano, C. Irvine, and G. Smith. A sound type system for secure flow analysis. *J. Comput. Secur.*, 4(2-3):167–187, 1996.

[VSS05] K. N. Verma, H. Seidl, and Th. Schwentick. On the complexity of equational Horn clauses. In *22th International Conference on Automated Deduction (CADE 2005)*, volume 3632 of *LNCS*, pages 337–352. Springer-Verlag, 2005.

[War05] Bogdan Warinschi. A computational analysis of the Needham-Schroeder-(Lowe) protocol. *J. Comput. Secur.*, 13(3):565–591, 2005.

[Wei99] Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *16th International Conference on Automated Deduction (CADE-16)*, volume 1632 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 1999.

[WL94] Thomas Y. C. Woo and Simon S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 28(3):24–37, 1994.

[ZM01] S. Zdancewic and A. Myers. Robust declassification. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 15–23, 2001.