Deciding key cycles for security protocols [⋆]

Véronique Cortier[1] and Eugen Zălinescu[1]

Loria UMR 7503 & INRIA Lorraine projet Cassis & CNRS, France

Abstract. Many recent results are concerned with interpreting proofs of security done in symbolic models in the more detailed models of computational cryptography. In the case of symmetric encryption, these results stringently demand that no key cycle (e.g. $\{k\}_k$) can be produced during the execution of protocols. While security properties like secrecy or authentication have been proved decidable for many interesting classes of protocols, the automatic detection of key cycles has not been studied so far.

In this paper, we prove that deciding the existence of key-cycles is NP-complete for a bounded number of sessions. Next, we observe that the techniques that we use are of more general interest and apply them to reprove the decidability of a significant existing fragment of protocols with timestamps.

1 Introduction

Security protocols are small programs that aim at securing communications over a public network like Internet. The design of such protocols is difficult and error-prone; many attacks are discovered even several years after the publication of a protocol. Two distinct approaches for the rigorous design and analysis of cryptographic protocols have been pursued in the literature: the so-called Dolev-Yao, symbolic, or formal approach on the one hand and the cryptographic, computational, or concrete approach on the other hand. In the symbolic approach, messages are modeled as formal terms that the adversary can manipulate using a fixed set of operations. The main advantage of this approach is its relative simplicity which makes it amenable to automated analysis tools (see, e.g., [7, 3, 22]). In the cryptographic approach, messages are bit strings and the adversary is an arbitrary probabilistic polynomial-time Turing machine. While results in this model yield strong security guarantees, the proofs are often quite involved and only rarely suitable for automation (see, e.g., [16, 6]).

Starting with the seminal work of Abadi and Rogaway [1], recent results investigate the possibility of bridging the gap between the two approaches. The goal is to obtain the best of both worlds: simple, automated security proofs that entail strong security guarantees. The approach usually 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. For example, in the case of asymmetric encryption, it has been shown [21] that the *perfect encryption assumption* is a sound abstraction for IND-CCA2, which corresponds to a well-established security level. The perfect encryption assumption intuitively states that encryption is a black-box that can be opened

[⋆] This work has been partially supported by the ACI-SI Satin and the ARA SSIA Formacrypt.

only when one has the inverse key. Otherwise, no information can be learned from a ciphertext about the underlying plaintext.

However, it is not always sufficient to find the right cryptographic hypotheses. Formal 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 [4], the authors 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 [1, 18] require that no key cycles can be generated during the execution of a protocol. Key cycles are messages like $\mathrm{enc}(k, k)$ or $\mathrm{enc}(k_1, k_2), \mathrm{enc}(k_2, k_1)$ 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 [4, 17] the authors require that a key k_1 never encrypts a key k_2 generated before k_1.

Some authors circumvent the problem of key cycles by providing new security definitions for encryption that allow key cycles [2, 5]. 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.

Our main 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 intruder, for a bounded number of sessions. To the best of our knowledge, this problem has not been addressed before. We therefore provide a necessary component for automated tools used in proving strong, cryptographical security properties, using existing soundness results. Our result has been obtained following the classical approach of Rusinowitch-Turuani [25], revisited by Comon-Lundh [11, 13], where protocols are represented by constraint systems. Since this initial procedure is already implemented in Avispa [3] for deciding secrecy and authentication properties, we believe that our algorithm can be easily implemented since it can be adapted from the existing procedure.

Our second contribution is to provide a generic approach derived from [11, 13] to decide general security properties. Comon-Lundh showed that any constraint system can be transformed in (possibly several) much simpler constraint systems that are called *solved forms*. We show using (almost) the same transformation rules that, in order to verify a property on a constraint system, it is always sufficient to verify the property on the solved forms obtained after transformation. Compared to [11, 13], the framework is slightly extended since we consider sorted terms, symmetric and asymmetric encryption, pairing and signatures. We use this approach to first prove NP-completeness of the key-cycle problem but also to show co-NP-completeness of secrecy for protocols with timestamps. We actually retrieve a significant fragment of the decidable class identified by Bozga *et al* [9]. We believe our result can lead more easily to an implementation since, again, we only need to adapt the procedure implemented in Avispa [3] while Bozga *et al* have designed a completely new decision procedure, which *de facto* has not been implemented.

The messages and the intruder capabilities are modeled in Section 2. In Section 3.1, we define constraint systems and show how they can be used to express protocol executions. In Section 3.2, we define security properties and the notion of satisfiability of constraint systems. In 3.3, we explain how the satisfiability problem of any security property can be reduced to the satisfiability of the same problem but on simpler constraint systems. We show in Section 4.1 how this approach can be used to obtain our main result of NP-completeness of the generation of key cycles and in Section 4.2 how it can be used to derive NP-completeness for protocols with timestamps. Some concluding remarks about further work can be found in Section 5.

2 Messages and intruder capabilities

2.1 Syntax

Cryptographic primitives are represented by function symbols. More specifically, we consider the *signature* $(\mathcal{S}, \mathcal{F})$ made of a set of *sorts* $\mathcal{S} = \{s, s_1 \dots\}$ and a set of *symbols* $\mathcal{F} = \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}, \langle\,\rangle, \mathrm{pub}, \mathrm{priv}\}$ together with arities of the form $\mathrm{ar}(f) = s_1 \times s_2 \to s$ for the four first symbols and $\mathrm{ar}(f) = s \to s'$ for the two last ones. The symbol $\langle\,\rangle$ represents the pairing function. The terms $\mathrm{enc}(m, k)$ and $\mathrm{enca}(m, k)$ represent respectively the message m encrypted with the symmetric (resp. asymmetric) key k. The term $\mathrm{sign}(m, k)$ represents the message m signed by the key k. The terms $\mathrm{pub}(a)$ and $\mathrm{priv}(a)$ represent respectively the public and private keys of an agent a.

We fix an infinite set of *names* $\mathcal{N} = \{a, b \dots\}$ and an infinite set of *variables* $\mathcal{X} = \{x, y \dots\}$. We assume that names and variables are given with sorts. The set of *terms of sort* s is defined inductively by

$$\begin{array}{lll} t ::= & & \text{term of sort } s \\ & \mid \; x & \quad \text{variable } x \text{ of sort } s \\ & \mid \; a & \quad \text{name } a \text{ of sort } s \\ & \mid \; f(t_1, \dots, t_k) & \quad \text{application of symbol } f \in \mathcal{F} \end{array}$$

where for the last case, we further require that t_i is a term of some sort s_i and $\mathrm{ar}(f) = s_1 \times \ldots \times s_k \to s$. We assume a special sort Msg that subsumes all the other sorts and such that any term is of sort Msg.

As usual, we write $\mathcal{V}(t)$ for the set of variables occurring in t. A term is *ground* or *closed* if and only if it has no variables. The *size* of a term t, denoted $|t|$, is defined inductively as usual: $|t| = 1$ if t is a variable or a name and $t = 1 + \sum_{i=1}^{n} |t_i|$ if $t = f(t_1, \dots, t_n)$ for $f \in \mathcal{F}$. If T is a set of terms then $|T|$ denotes the sum of the sizes of its elements. We denote by $St(t)$ the set of subterms of t.

Substitutions are written $\sigma = \{x_1 = t_1, \dots, x_n = t_n\}$ with $\mathrm{dom}(\sigma) = \{x_1, \dots, x_n\}$. We only consider *well-sorted* substitutions, that is substitutions for which x_i and t_i have the same sort. σ is *closed* if and only if all of the t_i are closed. The application of a substitution σ to a term t is written $\sigma(t) = t\sigma$.

$$\frac{S \vdash x \quad S \vdash y}{S \vdash \langle x, y\rangle} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{enc}(x, y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{enca}(x, y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{sign}(x, y)}$$

$$\frac{S \vdash \langle x, y\rangle}{S \vdash x} \qquad \frac{S \vdash \langle x, y\rangle}{S \vdash y} \qquad \frac{S \vdash \mathrm{enc}(x, y) \quad S \vdash y}{S \vdash x}$$

$$\frac{S \vdash \mathrm{enca}(x, \mathrm{pub}(y)) \quad S \vdash \mathrm{priv}(y)}{S \vdash x} \qquad \frac{S \vdash \mathrm{sign}(x, \mathrm{priv}(y))}{S \vdash x}\,\textit{(optional)} \qquad \frac{}{S \vdash x}\, x \in S$$

Fig. 1. Intruder deduction system.

Sorts are mostly left unspecified in this paper. They can be used in applications to express that certain operators can be applied only to some restricted terms. For example, sorts can be used to require that messages are encrypted only by atomic keys.

2.2 Intruder capabilities

The ability of the intruder is modeled by a deduction system described in Figure 1 and corresponds to the usual Dolev-Yao rules. The first line describes the *composition* rules, the two last lines describe the *decomposition* rules and the axiom. Intuitively, these deduction rules say that an intruder can compose messages by pairing, encrypting and signing messages provided he has the corresponding keys and conversely, it can decompose messages by projecting or decrypting provided it has the decryption keys. For signatures, the intruder is also able to *verify* whether a signature $\mathrm{sign}(m, k)$ and a message m match (provided she has the verification key), but this does not give her any new message. That is why this capability is not represented in the deduction system. We also consider an optional rule $\frac{S \vdash \mathrm{sign}(x, \mathrm{priv}(y))}{S \vdash x}$ that expresses that an intruder can retrieve the whole message from its signature. This property may or may not hold depending on the signature scheme, and that is why this rule is optional. Note that this rule is necessary for obtaining soundness properties w.r.t. cryptographic digital signatures. Our results hold in both cases (that is, when the deduction relation $\vdash$ is defined with or without this rule).

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

Example 1. The term $\langle k_1, k_2\rangle$ is deducible from the set $S_1 = \{\mathrm{enc}(k_1, k_2), k_2\}$. A proof of $S_1 \vdash \langle k_1, k_2\rangle$ is:

$$\frac{\dfrac{S_1 \vdash \mathrm{enc}(k_1, k_2) \quad S_1 \vdash k_2}{S_1 \vdash k_1} \qquad S_1 \vdash k_2}{S_1 \vdash \langle k_1, k_2\rangle}$$

3 Constraint systems and security properties

Constraint systems are quite common (see e.g. [13, 25]) in modeling security protocols. We recall here their formalism and show how they can be used to specify general security properties. Then we prove that any constraint system can be transformed into a simpler constraint system.

3.1 Constraint systems

Definition 1. *A* constraint system C *is a finite set of expressions* $T_i \Vdash \mathit{tt}$ *or* $T_i \Vdash u_i$*, where* T_i *is a non empty set of terms,* tt *is a special symbol that represents an always deducible term, and* u_i *is a term,* $1 \leq i \leq n$*, such that:*

- $T_i \subseteq T_{i+1}$*, for all* $1 \leq i \leq n-1$*;*
- *if* $x \in \mathcal{V}(T_i)$ *then* $\exists j < i$ *such that* $T_j = \min\{T \mid (T \Vdash u) \in C, x \in \mathcal{V}(u)\}$ *(for the inclusion relation) and* $T_j \subsetneq T_i$*.*

The *left-hand side* (*right-hand side*) of a constraint $T \Vdash u$ is T (respectively u). The *left-hand side* of a constraint system C, denoted by $\mathit{lhs}(C)$, is the maximal set of messages T_n. The *right-hand side* of a constraint system C, denoted by $\mathit{rhs}(C)$, is the set of right-hand sides of its constraints. $\mathcal{V}(C)$ denotes the set of variables occurring in C. $\perp$ denotes the unsatisfiable system. The *size* of a constraint system is defined as $|C| \overset{\mathsf{def}}{=} |\mathit{lhs}(C)| + |\mathit{rhs}(C)|$.

A constraint system is denoted as a conjunction of expressions. The left-hand side of a constraint system C usually represents the messages sent on the network.

Example 2. Consider the famous Needham-Schroeder asymmetric key authentication protocol [23] designed for mutual authentication.

$$\begin{array}{rl} A \to B: & \mathrm{enca}(\langle N_A, A\rangle, \mathrm{pub}(B)) \\ B \to A: & \mathrm{enca}(\langle N_A, N_B\rangle, \mathrm{pub}(A)) \\ A \to B: & \mathrm{enca}(N_B, \mathrm{pub}(B)) \end{array}$$

The agent A sends to B his name and a fresh nonce (a randomly generated value) encrypted with the public key of B. The agent B answers by copying A's nonce and adds a fresh nonce N_B, encrypted by A's public key. The agent A acknowledges by forwarding B's nonce encrypted by B's public key. We assume that a potential intruder has a complete control of the network: he may intercept, block and send new messages to arbitrary agents.

Let $T_0 = \{a, b, i, \mathrm{pub}(a), \mathrm{pub}(b), \mathrm{pub}(i), \mathrm{priv}(i)\}$ be the initial knowledge of the intruder. The following constraint system C_1 models 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. We consider this scenario for simplicity, but of course we could also consider for example A talking to B and B responding to I.

$$T_1 \overset{\mathsf{def}}{=} T_0 \cup \{\mathrm{enca}(\langle n_a, a\rangle, \mathrm{pub}(i))\} \Vdash \mathrm{enca}(\langle x, a\rangle, \mathrm{pub}(b)) \quad (1)$$

$$T_2 \overset{\mathsf{def}}{=} T_1 \cup \{\mathrm{enca}(\langle x, n_b\rangle, \mathrm{pub}(a))\} \Vdash \mathrm{enca}(\langle n_a, y\rangle, \mathrm{pub}(a)) \quad (2)$$

$$T_3 \overset{\mathsf{def}}{=} T_2 \cup \{\mathrm{enca}(y, \mathrm{pub}(i))\} \Vdash \mathrm{enca}(n_b, \mathrm{pub}(b)) \quad (3)$$

where n_a and n_b are names of sort Msg and x and y are variables of sort Msg. The set T_1 represents the messages known to the intruder once A has contacted the corrupted agent I. Then the equations 1 and 2 can be read as follows: if a message of the form $\mathrm{enca}(\langle x, a\rangle, \mathrm{pub}(b))$ can be sent on the network, then B would answer to this message by $\mathrm{enca}(\langle x, n_b\rangle, \mathrm{pub}(a))$, which is added to T_1. Subsequently, if a message of the form $\mathrm{enca}(\langle n_a, y\rangle, \mathrm{pub}(a))$ can be sent on the network, then A would answer by $\mathrm{enca}(y, \mathrm{pub}(i))$ since A believes she is talking to I. The run is successful if B can finish his session by receiving the message $\mathrm{enca}(n_b, \mathrm{pub}(b))$. Then B believes he has talked to A while A actually talked to I. If the protocol was secure, such a constraint system should not have a solution. The variables represent those parts of messages that are *a priori* unknown to the agents.

3.2 Security properties

A security property is modeled by a predicate on (lists of) terms. The terms represent some information about the execution of the protocol, for example the messages that are sent on the network.

Definition 2. *A* security property *is a predicate on lists of messages. For a list L we denote by L_s the set of messages of the list L.*

Let C be a constraint system, L a list of terms such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$ and P a security property. 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. Every substitution satisfies $T \Vdash \mathit{tt}$ and none satisfies $\bot$.*

Example 3. If the predicate P is simply the $\mathbf{true}$ predicate (which holds for any list of terms) and the only sort is Msg then we simply retrieve the usual constraint system deduction problem, which is known to be NP-complete [11, 25].

Example 4. Secrecy can be easily expressed by requiring that the secret data is not deducible from the messages sent on the network. We consider again the constraint system C_1 defined in Example 2. Let L_1 be a list of the messages in $\mathit{lhs}(C_1)$. We define the predicate P_1 to hold on a list of messages if and only if n_b is deducible from it. That is, $P_1(L) = \mathbf{true}$ iff $L_s \vdash n_b$. Then the substitution $\sigma_1 = \{x = n_a, y = n_b\}$ is a solution of C_1 for the property P_1 w.r.t. L_1 and corresponds to the attack found by G. Lowe [19]. Note that such a deduction-based property can be directly encoded in the constraint system by adding a constraint $T \Vdash n_b$ where $T = \mathit{lhs}(C_1)$.

Example 5. Authentication can also be defined using a predicate P_2 on lists of messages. For this purpose we use corresponding assertions and we introduce, following the syntax of Avispa [3], two new function symbols $\mathrm{witness}$ and $\mathrm{request}$ of arity 4 with the following intution: $\mathrm{request}(a, b, id, m)$ says that the agent a (id being simply a constant identifying the request since there might be several requests in one execution of a protocol) *now believes* that it is really agent b who sent the message m (that is, a authenticates b on m), and $\mathrm{witness}(b, a, id, m)$ says that b has just sent the message m to a. The predicate P_2 holds on a list L of messages if whenever $\mathrm{request}(a, b, id, m)$ appears in the list there is a corresponding occurrence $\mathrm{witness}(b, a, id, m)$ (defining an

R_1	$C \wedge T \Vdash u$	$\rightsquigarrow$	$C \wedge T \Vdash \mathit{tt}$	if $T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u$
R_2	$C \wedge T \Vdash u$	$\rightsquigarrow_\sigma$	$C\sigma \wedge T\sigma \Vdash u\sigma$	if $\sigma = \mathrm{mgu}(t, u)$, $t \in St(T)$, $t \neq u$, t, u not variables
R_3	$C \wedge T \Vdash u$	$\rightsquigarrow_\sigma$	$C\sigma \wedge T\sigma \Vdash u\sigma$	if $\sigma = \mathrm{mgu}(t_1, t_2)$, $t_1, t_2 \in St(T)$, $t_1 \neq t_2$, t_1, t_2 not variables
R_4	$C \wedge T \Vdash u$	$\rightsquigarrow$	$\perp$	if $\mathcal{V}(T, u) = \emptyset$ and $T \nvdash u$
R_f	$C \wedge T \Vdash f(u, v)$	$\rightsquigarrow$	$C \wedge T \Vdash u \wedge T \Vdash v$	for $f \in \{\langle\,\rangle, \mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}$

Fig. 2. Simplification rules.

injection) appearing before it in the list (that is, at a smaller position), for any agents a, b different from the intruder. Choosing L_2 to be a list of the messages in $lhs(C)$ following the order induced by the constraints (that is m appears before m' in L_2 whenever $m \in T_i$, $m \notin T_j$, $m' \in T_j$, $T_i \subseteq T_j$) we obtain Lowe's definition of injective agreement [20]. Formally, a protocol has an attack on the authentication property iff the constraint system C has a solution for $\overline{P_2}$ w.r.t. L_2, where $\overline{P_2}$ is the negation of P_2.

Consider again the constraint system C_1 defined in Example 2 where T_0 is replaced by $T_0' = T_0 \cup \{\mathrm{enca}(\langle n_a', a\rangle, \mathrm{pub}(b))\}$: the agent A also initiates a session with B, and $T_1' = T_0' \cup T_1 \cup \{\mathrm{witness}(b, a, 1, x), \mathrm{request}(b, a, 2, n_b)\}$: B asserts that A should rely on the value of x for his authentication to A, and now B believes he talked with A. The substitution σ_1 defined in Example 4 is a solution of C_1 for the property $\overline{P_2}$ w.r.t. L_2, since there is no corresponding witness assertion for $\mathrm{request}(b, a, 2, n_b)$ in L_2.

In Section 4, we provide other examples of predicates which encode time constraints or express that no key cycles are allowed.

3.3 General approach

Using some simplification rules, solving general constraint systems can be reduced to solving simpler constraint systems that we called solved. We say that a constraint system is *solved* if it is different from $\perp$ and each of its constraints are of the form $T \Vdash \mathit{tt}$ or $T \Vdash x$, where x is a variable. This corresponds to the notion of solved form in [13].

Solved constraint systems with the single sort Msg are particularly simple in the case of the $\mathtt{true}$ predicate since they always have a solution, as noticed in [11]. 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 $T \vdash x\theta = t$ for any constraint $T \Vdash x$ of the solved system.

The *simplification rules* we consider are defined in Figure 2. All the rules are in fact indexed by a substitution: when there is no index then the identity substitution is implicitly considered. We write $C \rightsquigarrow_\sigma^n C'$ 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 $C \rightsquigarrow_\sigma^n C'$ for some $n \geq 1$, or if $C' = C$ and σ is the empty substitution.

The simplification rules are correct, complete and terminating in polynomial time.

Theorem 1. *Let C be a constraint system, θ a substitution, P a security property and L a list of messages such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$.*

1. *(Correctness) If $C \leadsto_\sigma^* C'$ 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' and substitutions σ, θ' such that $\theta = \sigma\theta'$, $C \leadsto_\sigma^* C'$ and θ' is a solution of C' for the property P w.r.t. $L\sigma$.*
3. *(Termination) If $C \leadsto_\sigma^n C'$ for some constraint system C' and some substitution σ then n is polynomially bounded in the size of C.*

Theorem 1 extends the result of [11] to sorted messages and general security properties. This last point simply relies on the fact that whenever $C \leadsto_\sigma^* C'$ then $L(\sigma\theta) = (L\sigma)\theta$ for any substitution θ. We introduced explicit sorts since soundness results are usually obtained for models with atomic sorts for keys and nonces for example. The proof is actually a simple extension of [11] and all the details can be found in [14].

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 1. *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.*

4 Decidability of some specialized 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 [9] for protocols with timestamps.

4.1 Encryption cycles

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

In this section we restrict our attention to key cycles and key order on symmetric keys since there are very few papers constraining the key relations in an asymmetric setting. We consider atomic keys for symmetric encryption since soundness results are only obtained in this setting. In particular, there exists no general definition (with a cryptographic interpretation) of key cycles in the case of arbitrary composed keys. More precisely, we assume a sort $\mathsf{Key} \subset \mathsf{Msg}$ and we assume that the sort of enc is $\mathsf{Msg} \times \mathsf{Key} \rightarrow \mathsf{Msg}$. All the other symbols are of sort $\mathsf{Msg} \times \cdots \times \mathsf{Msg} \rightarrow \mathsf{Msg}$. Hence only names and variables can be of sort Key.

Key cycles Many definitions of key cycles are available in the literature. They are defined by considering cycles in the *encrypts* relation between keys. But this relation differs from one article to another. For example, the early definition proposed by Abadi and Rogaway [1] says that k encrypts k' as soon as there exists a term $\mathrm{enc}(m, k)$ such that k' is a subterm of m. For example, both $\mathrm{enc}(k, k)$ and $\mathrm{enc}(\mathrm{enc}(a, k), k)$ contain key cycles. However, in the definition proposed by Laud [18], $\mathrm{enc}(\mathrm{enc}(a, k), k)$ does not contain a key cycle since the relation "k encrypts k'" is restricted to keys k' that occur in plaintext. We consider the two variants for the notion of key cycles.

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 $s \sqsubseteq \mathrm{enc}(t, t')$. Intuitively, $s \sqsubseteq t$ if s is a subterm of t that occurs (at least once) in a plaintext position.

Definition 3. *Let ρ_1 be a relation chosen in $\{<_{st}, \sqsubseteq\}$. Let S be a set of messages and k, k' two terms of sort* Key. *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*

$$k'\,\rho_1\,m' \textit{ and } \mathrm{enc}(m', k) \sqsubseteq m.$$

With $\rho_1 = <_{st}$, we retrieve the definition of Abadi and Rogaway. With $\rho_1 = \sqsubseteq$ we retrieve the definition of Laud. For simplicity, we may write ρ_e instead of ρ_e^S if S is clear from the context.

We say that a set of messages S *contains a key cycle* if there is a cycle in the relation ρ_e^S. If m is a message we denote by ρ_e^m the relation $\rho_e^{\{m\}}$ and say that m contains a cycle if $\{m\}$ contains a cycle.

Definition 4. *Let K be a set of names of sort* Key. *We define the predicate P_{kc}^K as follows: P_{kc}^K holds on a list of messages L if and only if $S = L_s \cup \{m \mid L_s \vdash m\}$ contains a key cycle $(k_1, \ldots, k_n)$, with $n \geq 1$ and $k_i \in K$ for all $1 \leq i \leq n$.*

Definition of key order 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 [4] and [17], 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 5. *Let $\leq$ be a partial order on a set of names K of sort* Key. *We define the predicate $P_{\leq}^K$ as follows: $P_{\leq}^K$ holds on a list of messages L if and only if the encrypts relation ρ_e^S (restricted to $K \times K$), where $S = L_s \cup \{m \mid L_s \vdash m\}$, is* compatible *with $\leq$, that is*

$$k\,\rho_e^m\,k' \;\Rightarrow\; k' \not\leq k, \textit{ for all } k, k' \in K.$$

For example, in [4, 17] the authors choose $\leq$ to be the order in which the keys are generated: $k \leq k'$ if k has been generated before k'. We denote by $\overline{P}_{\leq}^K$ the negation of

$P_{\leq}^{K}$. Indeed, an attack in this context is an execution such that the encrypts relation is incompatible with $\leq$, that is the predicate $\overline{P}_{\leq}^{K}$ 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 1. *Let L be a list of messages, K a set of names of sort* Key *and $\leq$ a partial order on K. The predicate P_{kc}^{K} (resp. $\overline{P}_{\leq}^{K}$) holds on L if and only if*

– *there is $k \in K$ such that k is deducible from L_s, that is $L_s \vdash k$, or*
– *L_s contains a key cycle (resp. the encrypts relation on L_s is not compatible with $\leq$).*

Indeed if k is deducible from L_s then $\mathrm{enc}(k, k)$ is deducible from L_s. Hence there is a deducible message containing a key cycle and for which the encrypts relation is not compatible with the order $\leq$. If there are no deducible keys then it can be easily shown that the encrypts relation on any deducible message is included in the encrypts relation on L_s, hence the equivalences in the proposition.

Decidability In what follows, a solution of C for the $\mathtt{true}$ predicate w.r.t. an arbitrary list is said to be a *partial* solution to C. A *key* is a term of sort Key.

We show how to decide the existence of key cycles or the conformation to an order in polynomial time for solved constraint systems without variables of sort Key. Indeed, the instantiation of key variables can be guessed in advance (see the next section). Note that the set of messages on which our two predicates are applied usually contains all messages sent on the network and possibly some additional intruder knowledge.

Proposition 2. *Let C be a solved constraint system without variables of sort* Key *and L be a list of messages such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$ and $lhs(C) \subseteq L_s$. Let K be a set of names of sort* Key *such that $L_s\theta \not\vdash k$ for any θ partial solution of C and for any $k \in K$. Let $\leq$ be a partial order on K.*

– *Deciding whether C has a solution for P_{kc}^{K} w.r.t. L can be done in $\mathcal{O}(|L|+|K|^2)$.*
– *Deciding whether C has a solution for $\overline{P}_{\leq}^{K}$ w.r.t. L can be done in $\mathcal{O}(|L|+|K|^2)$.*

Since the keys of K are not deducible from $L_s\theta$, for any θ partial solution of C, we know by Proposition 1 that it is sufficient to look at the encrypts relation only on $L_s\theta$ (and not on every deducible term).

Since C is solved, any constraint of C is of the form $T \Vdash x$ or $T \Vdash \mathit{tt}$. For each variable x of C we denote $T_x = \min\{T \mid T \Vdash x \in C\}$. Let t_x be the term obtained by pairing all terms of T_x (in some arbitrary order). We construct the following substitution $\tau = \tau_1 \ldots \tau_q$, where q is the number of variables in C, and τ_j is defined inductively as follows:

- $\mathrm{dom}(\tau_1) = \{x_1\}$ and $x_1\tau_1 = t_{x_1}$
- $\tau_{j+1} = \tau_j \cup \{{}^{t_{x_i}\tau_j}/{}_{x_i}\}$, where $i = \min\{i' \mid x_{i'} \notin \mathrm{dom}(\tau_j)\}$.

The construction is correct by the definition of a constraint system. It is clear that τ is a partial solution of C.

We construct the following directed graph $G = (K, A)$ as follows: if k encrypts k' in L_s then $(k, k') \in A$; and, if k encrypts x in L_s (where $x \in \mathcal{V}(L_s)$) then $(k, k') \in A$ for all $k' \, \rho_1 \, x\tau$. The graph captures exactly the encrypts relation induced by τ on C and any possible encrypts relation is contained in the graph.

Lemma 1. *Let θ be a partial solution of C and $k, k' \in K$ be two non deducible keys, that is $L_s\theta \not\vdash k, k'$. If $k \, \rho_e^{L_s\theta} \, k'$, that is k encrypts k' in $L_s\theta$ then $(k, k') \in A$. Conversely, if $(k, k') \in A$ then $k \, \rho_e^{L_s\tau} \, k'$, that is k encrypts k' in $L_s\tau$.*

We deduce that deciding whether C has a solution for P_{kc}^K w.r.t. L can be done simply by deciding whether the graph G has cycles. Indeed, if there is a solution θ such that there is a cycle $(k_1, \ldots, k_n)$ in $L_s\theta$, that is, for each i we have $k_i \rho_e^{L_s\theta} k_{i+1}$, then (by Lemma 1) $(k_i, k_{i+1}) \in A$ for each i, that is $(k_1, \ldots, k_n)$ is a cycle in the graph G. Conversely, if $(k_1, \ldots, k_n)$ is a cycle in G then, again by Lemma 1, τ is a solution of C for P_{kc}^K w.r.t. L.

Deciding whether C has a solution for $\overline{P}_{\leq}^K$ w.r.t. L can be done by deciding whether the graph G has the following property P_G: there is $(k, k') \in A$ such that $k \leq k'$. And indeed, if there is a solution θ such that $\overline{P}_{\leq}^K$ holds on $L\theta$, that is $P_{\leq}^K$ does not hold on $L\theta$, then there are $k, k' \in K$ such that $k \rho_e^{L_s\theta} k'$ and $k \leq k'$. Hence $(k, k') \in A$ and $k \leq k'$. That is the property P_G on the graph G holds. Conversely, if the property P_G holds then there are $k, k' \in K$ such that k encrypts k' in $L\tau$ and $k \leq k'$, that is τ is a solution of C for $\overline{P}_{\leq}^K$ w.r.t. L.

The graph can be constructed in $\mathcal{O}(|L_s| + |K|^2)$. Testing for cycles and verifying property P_G can be simply done by traversing the graph in $\mathcal{O}(|K|^2)$.

NP-completeness Consider a constraint system C, a set K of names of sort Key and a list of messages L such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$ and $lhs(C) \subseteq L_s$. We want to decide the existence of a solution of C for P_{kc}^K (resp. $P_{\leq}^K$) w.r.t. L. By Proposition 1, there is a solution if and only if

1. either there exists $k \in K$ such that there exists a partial solution to $C_k \overset{\text{def}}{=} C \wedge L_s \Vdash k$,
2. or no key from $k \in K$ is deducible (that is $L_s\theta \not\vdash k$, for all θ partial solution of C) and C has a solution for P_{kc}^K (resp. $P_{\leq}^K$) w.r.t. L.

We guess whether we are in case 1 or 2 and in case 1 we also guess which key is deducible. In the first case, we check whether C_k has a partial solution in non-deterministic polynomial time using Corollary 1. In the second case, we guess an instantiation θ of variables of sort Key and of codomain the set of keys appearing in L (a finite set). Then we check whether $C\theta$ has a solution for P_{kc}^K (resp. $P_{\leq}^K$) w.r.t. $L\theta$ using Theorem 1 and Proposition 2.

NP-hardness is obtained by adapting the construction for NP-hardness provided in [25]. 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 key cycle if and only if the formula is satisfiable. The construction is the same as in [25] (pages 15 and 16) except that, in the last rule, the participant responds with the term $\mathrm{enc}(k, k)$, 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 [25], to the satisfiability of the corresponding 3SAT formula.

4.2 Timestamps

For modeling timestamps, we introduce a new sort $\mathsf{Time} \subseteq \mathsf{Msg}$ for time and we assume an infinite number of names of sort Time, represented by rational numbers or integers. 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 $\frac{}{S \vdash a}$ for any name a 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 $\sigma = \{x_1 = c_1, \ldots, x_k = c_k\}$, 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 6. *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 ($x_1 \geq x_2 + 30$).

Example 6. We consider the Wide Mouthed Frog Protocol [10].

$$\begin{array}{l} A \rightarrow S : A, \mathrm{enc}(\langle T_a, B, K_{ab}\rangle, K_{as}) \\ S \rightarrow B : \mathrm{enc}(\langle T_s, A, K_{ab}\rangle, K_{bs}) \end{array}$$

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 this timestamp is older than any other message he has received from S. As explained in [10], 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:

$$S_1 \overset{\mathsf{def}}{=} \{a, b, \langle a, \mathrm{enc}(\langle 0, b, k_{ab}\rangle, k_{as})\rangle\} \Vdash \langle a, \mathrm{enc}(\langle x_{t_1}, b, y_1\rangle, k_{as})\rangle, x_{t_2} \tag{4}$$

$$S_2 \overset{\mathsf{def}}{=} S_1 \cup \{\mathrm{enc}(\langle x_{t_2}, a, y_1\rangle, k_{bs})\} \Vdash \langle b, \mathrm{enc}(\langle x_{t_3}, a, y_2\rangle, k_{bs})\rangle, x_{t_4} \tag{5}$$

$$S_3 \overset{\mathsf{def}}{=} S_2 \cup \{\mathrm{enc}(\langle x_{t_4}, b, y_2\rangle, k_{as})\} \Vdash \langle a, \mathrm{enc}(\langle x_{t_5}, b, y_3\rangle, k_{as})\rangle, x_{t_6} \tag{6}$$

$$S_4 \overset{\mathsf{def}}{=} S_3 \cup \{\mathrm{enc}(\langle x_{t_6}, a, y_3\rangle, k_{bs})\} \Vdash \mathrm{enc}(\langle x_{t_7}, a, k_{ab}\rangle, k_{bs}) \tag{7}$$

where y_1, y_2, y_3 are variables of sort Msg and $x_{t_1}, \ldots, x_{t_7}$ 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 names 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 $\sigma = \{y_1 = y_2 = y_3 = y_4 = k_{ab}, x_{t_1} = 0, x_{t_2} = x_{t_3} = 30, x_{t_4} = x_{t_5} = 60, x_{t_6} = x_{t_7} = 90\}$.

Proposition 3. *Any timed property can be decided in non-deterministic polynomial time on solved constraints.*

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

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 [26]. 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.

NP-completeness We deduce by combining Theorem 1 and Proposition 2 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.

5 Further work

We have shown how the generic approach we have derived from [11, 13, 25] can be used to retrieve two NP-completeness results. The first one enables us to detect key cycles while the second one enables us to solve constraint systems with timed constraints. In both cases, we had to provide a decision procedure only for a simple class of constraint systems. Since the constraint-based approach [11, 13, 25] has already been implemented in Avispa [3], we plan, using our results, to adapt this implementation to the case of key cycles and timestamps.

More generally, taking advantage of our generic approach, we would like to explore how decision procedures of distinct security properties can be combined on solved constraint systems. In addition, in our two cases, decidability on solved constraints systems was quite simple. It would be interesting to understand which classes of properties can be decided in the same manner.

Regarding key cycles, our approach is valid for a bounded number of sessions only. Secrecy is undecidable in general [15] 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 [24, 12, 8, 27] for secrecy and an unbounded number of sessions. We plan to investigate how such fragments could be used to decide key cycles.

Acknowledgment. We are particularly grateful to Michaël Rusinowitch and Bogdan Warinschi for their very helpful suggestions.

References

1. M. Abadi and Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 2:103–127, 2002.
2. 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.
3. 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.
4. 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.
5. M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks. Cryptology ePrint Archive, Report 2005/421, 2005.
6. 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.
7. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96, 2001.

8. 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*, April 2003.
9. 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.
10. J. Clark and J. Jacob. A survey of authentication protocol literature. Available at `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.
11. 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`.
12. 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 *LNCS*, pages 148–164. Springer-Verlag, June 2003.
13. 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.
14. V. Cortier and E. Zălinescu. Deciding key cycles for security protocols, extended version. Available at `http://www.loria.fr/~zalinesc/papers/cz_keycycles.ps`.
15. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.
16. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
17. R. Janvier, Y. Lakhnech, and L. Mazare. (De)Compositions of Cryptographic Schemes and their Applications to Protocols. Cryptology ePrint Archive, Report 2005/020, 2005.
18. P. Laud. Encryption cycles and two views of cryptography. In *Nordic Workshop on Secure IT Systems (NORDSEC'02)*, 2002.
19. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, March 1996.
20. G. Lowe. A hierarchy of authentication specification. In *10th Computer Security Foundations Workshop (CSFW '97)*, pages 31–44, 1997.
21. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, pages 133–151, 2004.
22. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175, 2001.
23. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
24. R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, Mumbai, 2003.
25. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.
26. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.
27. K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational Horn clauses. In *Proc. of the 22th International Conference on Automated Deduction (CADE 2005)*, Lecture Notes in Computer Science, pages 337–352. Springer-Verlag, 2005.