Protocol Insecurity with Finite Number of Sessions is NP-complete

Michaël Rusinowitch and Mathieu Turuani
LORIA-INRIA- Université Henri Poincaré,
54506 Vandoeuvre-les-Nancy cedex, France
email:{rusi,turuani}@loria.fr

Abstract

We investigate the complexity of the protocol insecurity problem for a finite number of sessions (fixed number of interleaved runs). We show that this problem is NP-complete in a Dolev-Yao model of intruders. The result does not assume a limit on the size of messages and supports non-atomic symmetric encryption keys. We also prove that in order to build an attack with a fixed number of sessions the intruder needs only to forge messages of polynomial size, provided that they are represented as dags.

Introduction

Even assuming perfect cryptography, the design of protocols for secure electronic transactions is highly error-prone and conventional validation techniques based on informal arguments and/or testing are not sufficient for meeting the required security level.

On the other hand, verification tools based on formal methods have been quite successful in discovering new flaws in well-known security protocols. These methods include state exploration using model-checking as in [16, 25, 22, 5, 2], logic programming [19], term rewriting [6, 15], tree automata [12] or combination of these techniques. Other approaches aim at proving the correctness of a protocol . They are based on authentication logics or proving security property by induction using interactive proof-assistants (see [3, 23]).

Although the general verification problem is undecidable [11] even in the restricted case where the size of messages is bounded [10], it is interesting to investigate decidable fragments of the underlying logics and their complexity. The success of the practical verification tools indicates that there may exist interesting decidable fragments that capture many concrete security problems. For instance, [10] shows that when messages are bounded and when no nonces (i.e. new data) are created by the protocol and the intruder, then the existence of a secrecy flaw is decidable and DEXPTIME-complete. The complexity for the case of finite sessions is mentioned as open in [10].

A related decidability result is presented in [14, 1]. The authors give a procedure for checking whether an unsafe state is reachable by the protocol. Their result holds for the case of finite sessions but with no bounds on the intruder messages. The detailed proof in [1] does not allow general messages (not just names) as encryption keys. The authors have not analyzed the complexity of their procedure[1].

Our result states that for a fixed number of interleaved protocol runs but with no bounds on the intruder messages the existence of an attack is NP-complete. We allow public key encryption as well as the possibility of symmetric encryption with any message. In this paper we only consider *secrecy* properties. However *authentication* can be handled in a similar way. Hence, here a protocol is considered insecure if it is possible to reach a state where a secret term gets possessed by the intruder. Thanks to the proof technique, we have been able to extend the result directly to various intruder models and to protocols with choice points.

Layout of the paper: We first introduce in Section 1 our model of protocols and intruder and the notion of *normal attack*. Then in Section 2 we study properties of derivations with intruder rules. This allows us to derive polynomial bounds for normal attacks and then to show that the problem of finding a normal attack is in NP. We show in Section 3 that the existence of an attack is NP-hard. In Appendix we show that the NP procedure of Section 2 can be extended to handle stronger intruder model (Appendix 4.1) weaker intruder model (Appendix 4.2) and also protocols with choice points (Appendix 4.3).

[1] They have announced recently an NP procedure for atomic keys

1 The Protocol Model

We consider a model of protocols in the style of [4]. The actions of any honest principal are specified as a partially ordered list that associates to (the format of) a received message its corresponding reply. The activity of the intruder is modeled by rewrite rules on sets of messages. We consider that the initialization phase of distributing keys and other information between principals is implicit. The approach is quite natural and it is simple to compile a wide range of protocol descriptions to our formalism. For instance existing tools such as CAPSL [7, 8] or CASRUL [15] would perform this translation with few modifications.

We present our model more formally now.

Names and Messages

The messages exchanged during the protocol execution are built using pairing $\langle \_,\_ \rangle$ and encryption operators $\{\_\}^s_\_$, $\{\_\}^p_\_$. We add a superscript to distinguish between public key (^{p}) and symmetric key (^{s}) encryptions. The set of basic messages is finite and denoted by $Atoms$. It contains names for principals and atomic keys from the set $Keys$. Since we have a finite number of sessions we also assume any nonce is a basic message: we consider that it has been created before the session and belongs to the initial knowledge of the principal that generates it.

Any message can be used as a key for symmetric encryption. Only elements from $Keys$ are used for public key encryption. Given a public key (resp. private key) k, k^{-1} denotes the associated private key (resp. public key) and it is an element of $Keys$. Given a symmetric key k then, k^{-1} will denote the same key.

The messages are then generated by the following (tree) grammar:

$$\begin{array}{rl} msg ::= & Atoms \mid \langle msg, msg \rangle \\ & \mid \{msg\}^p_{Keys} \mid \{msg\}^s_{msg} \end{array}$$

For concision we denote by $m_1, m_2, \ldots, m_n$ the set of messages $\{m_1, m_2, \ldots, m_n\}$. Given two sets of messages M and M' we denote by M, M' the union of their elements and given a set of messages M and a message t, we denote by M, t the set $M \cup \{t\}$.

Protocol Specification

We shall describe protocols by a list of actions for each principal. In order to describe the protocol steps we introduce message terms (or terms in short). We assume that we have a finite set of variables Var. Then the set of terms is generated by the following tree grammar:

$$\begin{array}{rl} term ::= & Var \mid Atoms \mid \langle term, term \rangle \\ & \mid \{term\}^p_{Keys} \mid \{term\}^s_{term} \end{array}$$

Let $Var(t)$ be the set of variables that occur in a term t. A *substitution* assigns terms to variables. A *ground substitution* assigns messages to variables. The application of a substitution σ to a term t is written $t\sigma$. We also write $[x \leftarrow u]$ the substitution σ defined by $\sigma(x) = u$ and $\sigma(y) = y$ for $y \neq x$. The set of subterms of t is denoted by $Sub(t)$. These notations are extended to sets of terms E in a standard way. For instance, $E\sigma = \{t\sigma \mid t \in E\}$.

A principal (except the initiator) replies after receiving a message matching a specified term associated to its current state. Then from the previously received messages (and initial knowledge) he builds the next message he will send. This construction amounts to substitute values for the variables of another specified term.

A protocol is given with a finite set of principal names $Names \subseteq Atoms$, and a partially ordered list of steps for each principal name. This partial order is to ensure that the actions of each principal are performed in the right order. More formally we associate to each principal A a partially ordered finite set $(W_A, <_{W_A})$. Each protocol step is specified by a couple of terms denoted $R \Rightarrow S$ and is intended to represent some message R expected by a principal A and his reply S to this message. Hence a protocol specification P is given by:

$$\{(\iota, R_\iota \Rightarrow S_\iota) \mid \iota \in \mathcal{I}\}$$

where $\mathcal{I} = \{(A, i) \mid A \in Names \text{ and } i \in W_A\}$. We write $|\mathcal{I}|$ for the size of $\mathcal{I}$. $Init$ and End are fixed messages used to initiate and close a protocol session. An **environment** for a protocol is a set of messages. A **correct execution order** π is a one-to-one mapping $\pi : \mathcal{I} \to \{1, .., |\mathcal{I}|\}$ such that for all $A \in Names$ and $i <_{W_A} j$ we have $\pi(A, i) < \pi(A, j)$. In other words π defines an execution order for the protocol steps. This order is compatible with the partial order of each principal. A **protocol execution** is given by a ground substitution σ, a correct execution order π and a sequence of environments $E_0, .., E_{|\mathcal{I}|}$ verifying: $Init \in E_0$, $End \in E_{|\mathcal{I}|}$, and for all $1 \leq k \leq |\mathcal{I}|$, $R_{\pi^{-1}(k)}\sigma \in E_{k-1}$ and $S_{\pi^{-1}(k)}\sigma \in E_k$.

Each step ι of the protocol extends the current environment by adding the corresponding message $S_\iota\sigma$ when $R_\iota\sigma$ is present.

One can remark that principals are not allowed to generate any new data. But this is not bedded since the number of sessions is finite: New datas are no more than principals's initial knowledge.

Decomposition rules		Composition rules	
$L_d(\langle a,b\rangle):$	$\langle a,b\rangle \to a,b,\langle a,b\rangle$	$L_c(\langle a,b\rangle):$	$a,b \to a,b,\langle a,b\rangle$
$L_d(\{a\}^p_K):$	$\{a\}^p_K, K^{-1} \to \{a\}^p_K, K^{-1}, a$	$L_c(\{a\}^p_K):$	$a, K \to a, K, \{a\}^p_K$
$L_d(\{a\}^s_b):$	$\{a\}^s_b, b \to \{a\}^s_b, b, a$	$L_c(\{a\}^s_b):$	$a, b \to a, b, \{a\}^s_b$

Table 1. Intruder Rules (see Appendix for an extension)

Example: Needham Schroeder protocol

In the notation of [10], the Needham Schroeder protocol rules are as follows, when we assume that every nonce is included in the initial knowledge of the principal that will create it:

A1: $A_0(k_A)$
$\to A_1(k_A,k_B,N_A).N_{S1}(\{\langle N_A,k_A\rangle\}_{k_B})$
A2: $A_1(k_A,k_B,x_1).N_{R2}(\{\langle x_1,y_1\rangle\}_{k_A})$
$\to A_2(k_A,k_B,x_1,y_1).N_{S3}(\{y_1\}_{k_B})$
B1: $B_0(k_A).N_{R1}(\{\langle x_2,k_A\rangle\}_{k_B})$
$\to B_1(k_A,k_B,x_2,N_B).N_{S2}(\{\langle x_2,N_B\rangle\}_{k_A})$
B2: $B_1(k_A,k_B,x_3,y_3).N_{R3}(\{y_3\}_{k_B})$
$\to B_2(k_A,k_B,x_3,y_3)$

We can obtain the protocol steps for our setting from the rules of [10] by simply unifying all terms with the same root symbol A_i, for all principals A. Here, this means that $A_1(k_A,k_B,N_A)$ unifies with $A_1(k_A,k_B,x_1)$ and $B_1(k_A,k_B,x_2,N_B)$ unifies with $B_1(k_A,k_B,x_3,y_2)$. For Needham Schroeder protocol, we obtain:

((A,1), $Init \Rightarrow \{\langle N_A,K_A\rangle\}_{K_B}$)
((A,2), $\{\langle N_A,y_1\rangle\}_{K_A} \Rightarrow \{y_1\}_{K_B}$)
((B,1), $\{\langle x_2,K_A\rangle\}_{K_B} \Rightarrow \{\langle x_2,N_B\rangle\}_{K_A}$)
((B,2), $\{N_B\}_{K_B} \Rightarrow End$)

The orderings on steps are the ones that are expected: $W_A = W_B = \{1,2\}$ with $1 <_{W_A} 2$, $1 <_{W_B} 2$. Let us emphasize that unlike [10] in our specification we do not consider that the protocol specification as a set of rules where (free) variables can be replaced by any terms. In our case the scope of variables may include several lines in the specification. See for example the protocol Otway-Rees given in Appendix 4.4.

REMARK: In [10] notation some terms $AnnK(k_e^{-1})$ are used for public/private keys. But we do not need them here: since the possession of a key or a nonce by a principal is implicit through the actions he can perform.

Intruder

In the Dolev Yao model [9] the intruder has the ability to eavesdrop, to divert and memorize messages, to compose and decompose, to encrypt and decrypt when he has the key, to generate new messages and send them to other participants with a false identity. We assume here without loss of generality that the intruder systematically diverts messages, possibly modifies them and forwards them to the receiver under the identity of the official sender. In other words all communications are mediated by a hostile environment represented by the intruder. The intruder actions for modifying the messages are simulated by rewrite rules on sets of messages. The rewrite relation is defined by $M \to M'$ if there exists one of the rule $l \to r$ in the array below such that l is a subset of M and M' is obtained by replacing l by r in M. We write $\to^*$ for the reflexive and transitive closure of $\to$.

The set of messages S_0 represents the initial knowledge of the intruder. We assume that at least the name of the intruder $Charlie$ belongs to this set.

Intruder rules are divided in several groups: rules for composing or decomposing messages. These rewrite rules are the only one we consider in this paper and any mention of "rules" refer to *these* rules. In the following a, b and c represent any message and K represents any element of Key. For instance, the rule with label $L_c(\langle a,b\rangle)$ replaces a set of messages a,b by the following set of messages $a,b,\langle a,b\rangle$.

See Table 1 for complete the intruder rules, and Appendix for an extension. We denote the application of a rule R to a set E of messages with result E' by $E \to_R E'$. We write $L_c = \{L_c(a) |$ for all messages $a\}$, and L_d in the same way, and a is called the **principal term** of a rule $L_c(a)$ or $L_d(a)$. We call **derivation** a sequence of rule applications $E_0 \to_{R_1} E_1 \to_{R_2} .. \to_{R_n} E_n$. The rules R_i for $i = 1,..,n$ are called the rules of this derivation D. We write $R \in D$ (abusively) to denote that R is one of the rules R_i, for $i = 1,..,n$, that has been used in the derivation D.

One can remark that if the intruder was allowed to generate new data he won't be more powerful. He can already create infinitely many data only known to himself with simple encryptions. (For instance $\{N\}_N$, $\{\{N\}_N\}_N$, ... assuming that N is only known by the intruder)

Attacks

There is an attack in N protocol sessions if the intruder can obtain the secret term in its knowledge set after completing at most N sessions. We consider first the case of a single session. Then we shall sketch in Subsection 2.4 how to reduce the case of several sessions to the unique session case.

Since received messages are filtered by principals with the left-hand sides of protocol steps, the existence of an attack can be expressed as a constraint solving problem: is there a way for the intruder to build from its initial knowledge and already sent messages a new message (defined by a substitution for the variables of protocol steps) that will be accepted by the recipient, and so on, until the end of the session, and such that at the end the secret term is known by the intruder.

We introduce now a predicate $Forge$ for checking whether a message can be constructed by the intruder from some known messages. This predicate can be viewed as the combination of predicates $synth$ and $analz$ from L. Paulson [23].

Definition 1 (Forge) *Let E be a set of terms and let t be a term such that there is E' with $E \rightarrow^* E'$ and $t \in E'$. Then we say that t is* forged *from E and we denote it by $t \in Forge(E)$.*

We assume that there is a special message $Secret$ in the protocol specification. Let k be the cardinality of $\mathcal{I}$, i.e. the total number of steps of the protocol. An attack is a protocol execution where the intruder can modify each intermediate environment and where the message $Secret$ belongs to the final environment. In an attack the intruder is able to forge any message expected by a principal by using its initial knowledge and already sent messages (spied in the environments). This means, formally, that a given protocol execution, with sequence of environments $E_0, .., E_k$, is an attack if for all $1 \leq i \leq k$ we have $E_{i-1}, S_{\pi^{-1}(i)}\sigma \rightarrow^* E_i$ and $E_k, S_{\pi^{-1}(k)}\sigma \rightarrow^* E_{k+1}$ with $Secret \in E_{k+1}$. However by definition $t \in Forge(E)$ iff there is E' such that $E \rightarrow^* t, E'$. Hence we can reformulate the definition of an attack using the predicate $Forge$:

Definition 2 (attack) *Given a protocol $P = \{R'_\iota \Rightarrow S'_\iota \mid \iota \in \mathcal{I}\}$, a secret message $Secret$ and assuming the intruder has initial knowledge S_0, an* attack *is described by a ground substitution σ and a correct execution order $\pi : \mathcal{I} \longrightarrow 1, \ldots, k$ such that for all $i = 1, \ldots, k$, we have:*

$$R_i\sigma \in Forge(S_0, S_1\sigma, ..., S_{i-1}\sigma)$$
$$Secret \in Forge(S_0, S_1\sigma, ..., S_k\sigma)$$

where $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$.

We introduce now a measure on attacks and a notion of minimal attack among all attacks, called *normal attack*. We shall prove in the next sections that normal attacks have polynomial bounds for a suitable representation of terms.

The *size* of a message term t is denoted $|t|$ and defined as:

- $|Charlie| = 0$
- $|t| = 1$ for any other element $t \in Atoms$
- and recursively by $|\langle x, y\rangle| = |\{x\}_y| = |x| + |y| + 1$

Note that $Charlie$ is the minimal size message. We recall that a finite multiset over natural numbers is a function M from $\mathbb{N}$ to $\mathbb{N}$ with finite domain. We shall compare finite multisets of naturals by extending the ordering on $\mathbb{N}$ as follows: $M \gg N$ if $i)$ $M \neq N$ and $ii)$ whenever $N(x) > M(x)$ then $M(y) > N(y)$ for some $y > x$.

Definition 3 (normal attack) *An attack is* normal *if the multiset of nonnegative integers $\langle |R_1\sigma|, ..., |R_k\sigma|\rangle$ is minimal*

Clearly if there is an attack there is a normal attack since the measure is a well-founded ordering on finite multisets of nonnegative integers. We now present an NP procedure for detecting the existence of a normal attack.

2 Existence of a Normal Attack is in NP

We first show some basic facts on the representation of message terms as *Directed Acyclic Graph* (DAG). Then we shall show how to obtain from any derivation a more compact one. We will then be able to prove that a normal attack has a polynomial size w.r.t. the size of the protocol and intruder knowledge, when using DAG representations.

2.1 Preliminaries

The **DAG-representation** of a set E of message terms is the graph $(\mathcal{V}, \mathcal{E})$ with labeled edges, where:

- the set of vertices $\mathcal{V} = Sub(E)$, the set of subterms of E.
- the set of edges $\mathcal{E}$:

$$\left\{ \begin{array}{l} \{v_s \xrightarrow{left} v_e \mid \exists b,\ v_s = \{v_e\}_b \text{ or } v_s = \langle v_e, b\rangle\} \\ \cup\{v_s \xrightarrow{right} v_e \mid \exists b,\ v_s = \{b\}_{v_e} \text{ or } v_s = \langle b, v_e\rangle\} \end{array} \right.$$

Remark 1 *The DAG representation is unique.*

1. Guess a correct execution order $\pi : \mathcal{I} \longrightarrow \{1, ..., k\}$.
   Let $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$ for $i \in \{1, ..., k\}$
2. Guess a ground substitution such that for all $x \in V$, $\sigma(x)$ has DAG-size $\leq n$.
3. For each $i \in \{1, ..., k+1\}$ guess an ordered list l_i of n^2 rules whose principal terms have DAG-size $\leq 3.n^2$.
4. For each $i \in \{1, ..., k\}$ check that l_i applied to $\{S_j\sigma | j < i\} \cup \{S_0\}$ generates $R_i\sigma$
5. Check that l_{k+1} applied to $\{S_j\sigma | j < k+1\} \cup \{S_0\}$ generates $Secret$.
6. If each check is successful then answer YES.

Figure 1. NP procedure for the Insecurity Problem

If n is the number of elements in $Sub(t)$, one can remark that $(\mathcal{V}, \mathcal{E})$ has at most n nodes and $2.n$ edges. Hence its size is linear in n, and for convenience we shall define the DAG-size of E, denoted by $|E|_{DAG}$, to be the number of distinct subterms of E, i.e the number of elements in $Sub(E)$. For a term t, we simply write $|t|_{DAG}$ for $|\{t\}|_{DAG}$.

Lemma 1 *For all set of terms E, for all variable x and for all message t, we have: $|E[x \leftarrow t]|_{DAG} \leq |E, t|_{DAG}$*

Proof: see Appendix 4.5. □

Corollary 1 *For all set of terms E, for all ground substitution γ, we have $|E\gamma|_{DAG} \leq |E, \gamma(x_1), .., \gamma(x_k)|_{DAG}$ where $\{x_1, .., x_k\}$ is the set of variables in Var (recall that Var is finite) such that $\gamma(x_i) \neq x_i$.*

Proof: We just apply Lemma 1 above for each x_i. □

Remark 2 *To check that a rule $l \rightarrow l, r'$ applies to E and to compute the resulting DAG E', when we are given a DAG-representation of E, l and r', only polynomial time is required since one first checks that all terms in l are also in E and then one add r' and compute the DAG-representation E' of E, r'.*

We are going to present an NP decision procedure for finding an attack, assuming an attack exists. The procedure amounts to guess a correct execution order π, a possible ground substitutions σ, for variables Var, with a DAG-size that is polynomially bounded, then to guess k+1 lists of rules of length n^2 and finally to check that when applying these lists of rules the intruder can build all expected messages as well as the secret.

We assume given a protocol specification $\{(\iota, R'_\iota \Rightarrow S'_\iota) \mid \iota \in \mathcal{I}\}$. Let $P = \{R'_\iota, S'_\iota \mid \iota \in \mathcal{I}\}$, a secret message $Secret$ and a finite set of messages S_0 for initial intruder knowledge. If P, S_0 is not given in DAG-representation, they are first converted to this format (in polynomial time). We assume that the DAG-size of P, S_0 is n, the finite set of variables in P is V, and $|\mathcal{I}| = k$.

The NP procedure for checking the existence of an attack is written in Figure 1. To prove the correction of this procedure we shall show that we can put a bound on the length of normal attacks. We will first give properties about derivations. We will also give polynomial bounds on the substitution σ that is used in a normal attack.

2.2 Derivations

In this section, we will give some useful definitions and properties on derivations. We shall introduce a notion of normal derivation, denoted by $Deriv_t(E)$. A related notion of normal derivation has been studied in [5]. Rather than a natural deduction presentation in [5] we use here term rewriting.

Definition 4 *Given a derivation $D = E_0 \rightarrow_{R_1} E_1 \rightarrow_{R_2} .. \rightarrow_{R_n} E_n$, a term t is a goal of D if $t \in E_n$ and $t \notin E_{n-1}$.*

For instance if $t \in Forge(E)$ there exists a derivation with goal t: we take a derivation $D = E \rightarrow_{R_1} \ldots \rightarrow_{R_n} E'$ with $t \in E'$ and then we take the smallest prefix of D containing t.

This allows us to define some normal derivation, i.e. derivation minimal in length:

Definition 5 *We denote $Deriv_t(E)$ a derivation of minimal length among the derivations from E with goal t (chosen arbitrarily among the possible ones).*

In order to bound the length of such derivations, we can prove the two following lemmas: All intermediate terms in $Deriv_t(E)$ is a subterm of E or t.

Lemma 2 *If there exists t' such that $L_d(t') \in Deriv_t(E)$ then t' is a subterm of E*

Proof: Let $D = Deriv_t(E)$. By minimality of D, we have $L_c(t') \notin D$. Then either $t' \in E$ and we have the conclusion of the lemma. Otherwise there exists a rule $L_d(t_1[t'])$ in D generating t'. But any rule in D generating $t_1[t']$ must be in L_d (if not, the decomposition would be useless and the derivation would not be minimal): we can iterate this reasoning on $t_1[t']$, and this ends the proof: t' increases strictly at each iteration and the derivation only contains a finite number of terms. $\square$

Lemma 3 *If there exists t' such that $L_c(t') \in Deriv_t(E)$ then t' is a subterm of $\{t\} \cup E$*

Proof: Let $D = Deriv_t(E)$. By minimality of D, we have $L_d(t') \notin D$. Hence either $t' \in \{t\} \cup E$ and the lemma is proved. Otherwise there is at last one rule using t': if not, $L_c(t')$ would be useless and $Deriv_t(E)$ not minimal. Then one of the following cases can be applied:

- There exists a such that $L_d(\{a\}_{t'^{-1}}) \in D$, hence $\{a\}_{t'^{-1}}$ is a subterm of E by the Lemma 2, and so is t'.
- Or there exists b such that $L_c(\{t'\}_b) \in D$ or $L_c(\{b\}_{t'}) \in D$. In this case, we can iterate this reasoning on $t_1 = \{t'\}_b$ or $t_1 = \{b\}_{t'}$. This ends the proof, because t' strictly increases at each iteration and the derivation only contain a finite number of terms. $\square$

We show in the next proposition that there always exists derivations of a term t from a set E with a number of rules bounded by the DAG-size of initial and final terms t, E. This will be very useful to bound the length of the derivations involved in the research of an attack.

Proposition 1 *For any set of terms E and for any term t, if $Deriv_t(E) = E \rightarrow_{L_1} E_1 .. \rightarrow_{L_n} E_n$ then $n \leq |t, E|_{DAG}$ and for all $1 \leq i \leq n$, $|E_i|_{DAG} \leq |t, E|_{DAG}$.*

Proof: Let us prove that the number of steps in $Deriv_t(E)$ is at most $|t, E|_{DAG}$ by examining the terms composed or decomposed for any rule R that has been applied in $Deriv_t(E)$:

- From Lemma 2 every term decomposed (with L_d) is derived from E by decompositions exclusively. Hence every term which is decomposed was a subterm of E and is counted in $|E|_{DAG}$.
- From Lemma 3 every term composed (by L_c) is used as a subterm of a key or of t. Hence it is counted in $|t, E|_{DAG}$.
- Every rule R either composes or decomposes a term, but R never composes (resp. decomposes) a term which has already been composed (resp. decomposed). Hence to each subterm of E or t corresponds at most one rule application in $Deriv_t(E)$ for composing or decomposing it. (merging identical subterms)

Hence the number of terms composed or decomposed in $Deriv_t(E)$ is bounded by the number of distinct subterms of E, t and the first part of the result follows. Since each intermediate term is a subterm of E, t, the second part of the proposition follows. $\square$

An other kind of useful derivations is shown in the following Proposition 2. It will allow us to prove the Lemma 4.

Proposition 2 *Let $t \in Forge(E)$ and $\gamma \in Forge(E)$ be given with $Deriv_\gamma(E)$ ending with an application of a rule in L_c. Then there is a derivation D with goal t starting from E, and verifying* Condition 2*: $L_d(\gamma) \notin D$*

Proof: Let $t \in Forge(E)$ and $\gamma \in Forge(E)$ be with $Deriv_\gamma(E)$ ending with an application of a rule in L_c. Let D be $Deriv_\gamma(E)$ without it's last rule, i.e. $Deriv_\gamma(E)$ is D followed by L_c, and let D' be the derivation obtained from $Deriv_t(E)$ by replacing every decomposition L_d of γ by D. Then:

- D' is a correct derivation: since L generates α and β, the two distinct direct subterms of γ. (since γ is obtained by compositing with L_c)
- L does not contain a decomposition L_d of γ from the fact that $Deriv_\gamma(E)$ has γ as goal. Otherwise the last composition would be useless.
- Hence D' satisfies *Condition 2* and the lemma follows. $\square$

2.3 Polynomial Bounds on Normal Attacks

We shall prove that when there exists an attack then a normal attack can always be constructed from subterms that are already occurring in the problem specification. This will allow to give bounds on the messages size and on the number of rewritings involved in such an attack.

Hence let us assume a protocol $P = \{R'_\iota \Rightarrow S'_\iota \mid \iota \in \mathcal{I}\}$, a secret message $Secret$ and a set of messages S_0 as the initial intruder knowledge. We assume that there exists an attack described by a ground substitution σ and a correct execution order $\pi : \mathcal{I} \longrightarrow 1, \ldots, k$ (where k is the cardinality of $\mathcal{I}$). We define $R_i = R'_{\pi^{-1}}$ and $S_i = S'_{\pi^{-1}}$ for $i = 1, \ldots, k$.

We also define: $\mathcal{SP}$ as the set of subterms of the terms in the set $\mathcal{P} = \{R_j | j = 1, \ldots, k\} \cup \{S_j | j = 0, \ldots, k\}$, and $\mathcal{SP}_{\leq i}$ the set of subterms of the terms in $\{R_j | j = 1, \ldots, i\} \cup \{S_j | j = 0, \ldots, i\}$.

We assume without loss of generality that $Charlie \in S_0$ i.e. the intruder initially knows its name !

Definition 6 *Let t and t' be two terms and θ a ground substitution. Then t is a θ-match of t' if t is not a variable and $t\theta = t'$. This will be denoted by $t \sqsubseteq_\theta t'$*

The following lemma is one of the key property of this paper. It allows us to prove that every substitution σ in a normal attack is only build with parts of the protocol specification. In this way, we will be able to prove that all substitution σ in a normal attack has a DAG-size bounded by a polynomial in the protocol DAG-size.

Lemma 4 *Given a normal attack σ, for all variable x, there exists $t \sqsubseteq_\sigma \sigma(x)$ such that $t \in \mathcal{SP}$.*

Proof: Let σ be a normal attack, and let us first assume that there exists x such that for all t such that $t \sqsubseteq_\sigma \sigma(x)$ we have $t \notin \mathcal{SP}$, and let us derive a contradiction. Let us define:

$$N_x = min\{j / \sigma(x) \in \mathcal{SP}_{\leq j}\}$$

N_x is the first step of the protocol whose message contains $\sigma(x)$ as a subterm. However since for all t such that $t \sqsubseteq_\sigma \sigma(x)$, t is not in $\mathcal{SP}$, there exists a variable y which is subterm of R_{N_x} or S_{N_x} such that $\sigma(x)$ is a subterm of $\sigma(y)$. (Otherwise there would exist a σ-match of $\sigma(x)$ with some subterm of R_{N_x}). Then let us show now the following claim:

Claim $\sigma(x) \in Forge(S_0\sigma, ..., S_{N_x-1}\sigma)$.
proof: Let $Deriv_{R_{N_x}\sigma}(S_0\sigma, .., S_{N_x-1}\sigma)$ be $E_0 \rightarrow_{L_1} E_1 .. \rightarrow_{L_n} E_n$. Since $\sigma(x)$ is a subterm of R_{N_x} and since $R_{N_x}\sigma \in Forge(S_0\sigma, ..., S_{N_x-1}\sigma)$, we have:

- if there exist $i \leq n$ such that $\sigma(x) \in E_i$, then obviously $\sigma(x) \in Forge(S_0\sigma, ..., S_{N_x-1}\sigma)$.
- Otherwise, we will prove by induction that $\sigma(x)$ occurs as a subterm in every intermediary set E_i. We have $\sigma(x)$ subterm of E_n since $R_{N_x}\sigma \in E_n$, and:
  - If $\sigma(x)$ subterm of $s \in E_i$ and if there exists $j \leq i$ such that $L_j = L_c(s)$, then $s \neq \sigma(x)$ since $\sigma(x) \notin E_j$. Hence, $\sigma(x)$ is a subterm of E_{j-1}.
  - If $\sigma(x)$ subterm of $s \in E_i$ and if there exists $j \leq i$ s.t. s created by $L_j = L_d(r)$, then s and $\sigma(x)$ are subterms of E_{j-1}.
  - If such a rule L_j does not exist, then $\sigma(x)$ is a subterm of E_0 and the iteration is finished.

  This iteration implies that $\sigma(x)$ is a subterm of $E_0 = S_0\sigma, .., S_{N_x-1}\sigma$. But it's impossible due to the choice of N_x.

end

Hence there exists a derivation $Deriv_{\sigma(x)}(S_0\sigma, ..., S_{N_x-1}\sigma)$ and we can notice that its last step uses necessarily a composition rule since otherwise Lemma 2 would imply that $\sigma(x)$ is a subterm of $S_0\sigma, ..., S_{N_x-1}\sigma$, and therefore a contradiction.

Let us define the substitution σ' to be equal to σ on all variables except for x where $\sigma'(x) = Charlie$. We will prove that σ' defines an attack with the same execution order than σ. Since σ is an attack, for all j, $R_j\sigma \in Forge(S_0\sigma, .., S_{j-1}\sigma)$.

If $j < N_x$ then since $\sigma(x)$ has no occurrence in $R_j\sigma, S_0\sigma, .., S_{j-1}\sigma$, we have $R_j\sigma = R_j\sigma'$, $S_0\sigma = S_0\sigma'$, $\ldots$, $S_{j-1}\sigma = S_{j-1}\sigma'$. Therefore we also have $R_j\sigma' \in Forge(S_0\sigma', .., S_{j-1}\sigma')$

If $j \geq N_x$ then there exists a derivation with goal $R_j\sigma$

$$E_0 \rightarrow_{L_1} E_1 \rightarrow_{L_2} .... \rightarrow_{L_{n_j}} E_{n_j}$$
where $E_0 = S_0\sigma, .., S_{j-1}\sigma$

By Proposition 2, $\sigma(x)$ is never simplified or decomposed in this derivation: $\forall i \leq n_j$, $L_i \neq L_d(\sigma(x))$ and $L_i \neq L_d(\sigma(x))$. Let us build from this derivation a new one where each $\sigma(x)$ is replaced by $Charlie$. We shall denote by $t\delta$ the term obtained from t by replacing every occurrence of $\sigma(x)$ by $Charlie$. For convenience we shall consider that $E \rightarrow E$ is a derivation step justified by the identity rule $\emptyset$. Then we shall prove that there exists a valid derivation:

$$E_0\delta \rightarrow_{L'_1} E_1\delta \rightarrow_{L'_2} .... \rightarrow_{L'_{n_j}} E_{n_j}\delta$$

where every rule L'_i is either L_i or $\emptyset$. Hence we only have to take the same rules as in the initial derivation but possibly skip some steps. More precisely let us show that for $i = 1 \ldots n_j$ when $E_{i-1} \rightarrow_{L_i} E_i$ then either $E_{i-1}\delta \rightarrow_{L_i} E_i\delta$ or $E_{i-1}\delta \rightarrow_{\emptyset} E_i\delta$.

1. if $L_i = L_c(\langle \alpha, \beta \rangle)$ and:
   (a) if $\sigma(x) \neq \langle \alpha, \beta \rangle$, then $(E'_{i-1}, \alpha, \beta)\delta \rightarrow_{L_i} (E'_i, \alpha, \beta, \langle \alpha, \beta \rangle)\delta$ is a valid step since $\langle \alpha\delta, \beta\delta \rangle = \langle \alpha, \beta \rangle \delta$.
   (b) else $\sigma(x) = \langle \alpha, \beta \rangle$ and we can take $L'_i = \emptyset$ since $Charlie \in E_i$, for all i.
2. same reasoning for $L_i = L_c(\{\alpha\}_\beta)$.
3. if $L_i = L_d(\langle \alpha, \beta \rangle)$ then $\sigma(x) \neq \langle \alpha, \beta \rangle$, and $(E'_{i-1}, \langle \alpha, \beta \rangle)\delta \rightarrow_{L_i} (E'_i, \langle \alpha, \beta \rangle, \alpha, \beta)\delta$ is valid since $\langle \alpha\delta, \beta\delta \rangle = \langle \alpha, \beta \rangle \delta$.
4. same reasoning for $L_i = L_d(\{\alpha\}_\beta)$.

Finally we get for all j, $R_j\sigma' \in Forge(S_0\sigma', .., S_{j-1}\sigma')$. Hence it follows that σ' is an attack for the same protocol order than σ. Since σ' is obtained from σ by simply replacing the value of x by a strictly smaller one (w.r.t. $|\_|$) we have $\langle |R_1\sigma'|, ..., |R_k\sigma'| \rangle$ strictly smaller than $\langle |R_1\sigma|, ..., |R_k\sigma| \rangle$ and this is contradictory with the assumption of normal attack for σ. □

We can now use this lemma to bound the DAG-size of every $\sigma(x)$. This is shown in the following Theorem:

Theorem 1 *If σ is the substitution in a normal attack then we have for all $x \in Var$ $|\sigma(x)|_{DAG} \leq |\mathcal{P}|_{DAG}$*

Proof: Given a set of variable U, we shall write $\overline{U} = \{\sigma(x) \mid x \in U\}$. Let us build by induction a sequence of sets $E_p \subseteq \mathcal{SP}$ and a sequence of sets V_p of variables such that $|\sigma(x)|_{DAG} \leq \left|E_p, \overline{V_p}\right|_{DAG}$:

- Let (E_0, V_0) be $(\emptyset, \{x\})$. We have $|\sigma(x)|_{DAG} \leq \left|E_0, \overline{V_0}\right|_{DAG}$ and $E_0 \subseteq \mathcal{SP}$.
- Assume that we have built (E_p, V_p) such that $|\sigma(x)|_{DAG} \leq \left|E_p, \overline{V_p}\right|_{DAG}$ and $E_p \subseteq \mathcal{SP}$, let us define E_{p+1} and V_{p+1}:

  If $V_p \neq \emptyset$ let us choose $x' \in V_p$. Then there exists $t \sqsubseteq_\sigma \sigma(x')$ such that $t \in \mathcal{SP}$. We define $E_{p+1} = E_p \cup \{t\}$ and $V_{p+1} = Var(t) \cup V_p - \{x'\}$. Since $t \in \mathcal{SP}$, we have $E_{p+1} \subseteq \mathcal{SP}$. Let us show that $|\sigma(x)|_{DAG} \leq \left|E_{p+1}, \overline{V_{p+1}}\right|_{DAG}$. Let $\delta = \{[y \leftarrow \sigma(y)] \,/ y \in Var(t)\}$. By applying the Corollary 1 on $E_p \cup \{t\} \cup \overline{V_p - x'}$ for the substitution δ. (*Remark:* $t\delta = \sigma(x')$) We obtain:

  $$\left|E_p\delta, \overline{V_p}\right|_{DAG} \leq \left|E_p, \overline{V_p - x'}, t, \overline{Var(t)}\right|_{DAG}$$

  and then

  $$\left|E_p, \overline{V_p}\right|_{DAG} \leq \left|E_p\delta, \overline{V_p}\right|_{DAG} \leq \left|E_{p+1}, \overline{V_{p+1}}\right|_{DAG}$$

Finally, this construction terminates since $\sum_{y \in V_p} |\sigma(y)|$ strictly decreases. At the end we get $V_p = \emptyset$ and $|\sigma(x)|_{DAG} \leq |E_p|_{DAG}$ with $E_p \subseteq \mathcal{SP}$: since $|E_p|_{DAG} \leq |\mathcal{P}|_{DAG}$ this proves the theorem. □

The consequence of Theorem 1 is that the DAG-size of the messages that are sent or received during a normal attack is bounded by a polynomial in the DAG size of the protocol. This result has crucial practical implications since it means that when searching for an attack we can give a simple *a priori* bound on the dag-size of the messages needed to be forged by the intruder:

Corollary 2 *If σ is the substitution in a normal attack then for all $i = 1, \ldots, k$: $|R_i\sigma|_{DAG} \leq |\mathcal{P}|^2_{DAG}$ and $|S_i\sigma|_{DAG} \leq |\mathcal{P}|^2_{DAG}$.*

Proof: $|S_i\sigma|_{DAG} \leq |S_i, \{\sigma(x) | x \in Var(S_i)\}|_{DAG}$ by Corollary 1. From Lemma 4 we also have for all $x \in Var$, $|\sigma(x)| \leq |\mathcal{P}|_{DAG}$. Let $|Var|$ be the number of distinct variables in the protocol specification. Hence $|S_i\sigma|_{DAG} \leq |S_i|_{DAG} + (|\mathcal{P}|_{DAG}) \times |Var|$ and since $|S_i|_{DAG} \leq |\mathcal{P}|_{DAG}$ and $|Var| + 1 \leq |\mathcal{P}|_{DAG}$ we have immediately $|S_i\sigma|_{DAG} \leq |\mathcal{P}|^2_{DAG}$. The same proof holds for $R_i\sigma$. □

2.4 Protocol Insecurity with Finite Number of Sessions is in NP

We recall here the NP procedure for checking the existence of an attack and shows its correctness.

We assume given a protocol specification $\{(\iota, R'_\iota \Rightarrow S'_\iota) \mid \iota \in \mathcal{I}\}$. Let $P = \{R'_\iota, S'_\iota \mid \iota \in \mathcal{I}\}$, a secret message $Secret$ and a finite set of messages S_0 for initial intruder knowledge. If P, S_0 is not given in DAG-representation, they are first converted to this format (in polynomial time). We assume that the DAG-size of P, S_0 is n, the finite set of variables in P is V, and $\mathcal{I} = k$.

Let us first remark that the procedure written in Section 2.1 is NP:

- A correct execution order is a permutation of $\mathcal{I}$, and can be guessed in polynomial time.

- Since $\sigma(x)$ has DAG-size $\leq n$, one can choose a DAG representation of $\sigma(x)$ in time $O(n)$ and σ in $O(n^2)$.
- Since each rule in l_i has DAG-size $\leq n^2$ and there is n^2 rules, one can choose each l_i in time $O(n^4)$, and all l_i in time $O(n^5)$. *Remark:* each term in the rules is in DAG representation.
- Computing the result E' of the application of a rule $L_x(t)$ on E, with E and t in DAG representation, can be done in polynomial time in the DAG-size of E by Remark 2. But $|E'|_{DAG} \leq |E|_{DAG}+2$ (since any rule application introduces at most 2 symbols in E) and therefore each intermediate set of terms has DAG-size $\leq 3.n^2$, and each application rule takes polynomial time in n. So, checking that all l_i are correctly applied takes polynomial time of n. To verify that $R_i\sigma$ is in the last set of terms takes obviously polynomial time too.

We can now see that this procedure is correct: it answers YES if and only if the protocol has an attack:

- If an Attack exists then a Normal Attack exists: Since $D = Deriv_{R_i\sigma}(\{S_j\sigma | j < i\} \cup \{S_0\})$ has at most n^2 rules (Proposition1 and Theorem1), each intermediate set of terms has DAG-size $\leq 3.n^2$. Hence, each term involved in a rule in D has DAG-size $\leq 3.n^2$. Therefore D will be a possible guess for l_i (as well as the execution order associated to this attack).
- If the procedure answers YES, the checking performed on the guessed derivations proves than the protocol has an attack.

Multiple sessions: When considering several protocol sessions simultaneously we assume that they are independent (in the absence of intruder) and therefore one can assume that their specifications do not share variables.

In order to reduce the security problem for several sessions to the security problem for one session, one can simply compose the sessions into a unique session of a more complex protocol. This construction is sketched below.

Given two disjoint partially orders $<$ and $<'$ defined on disjoint sets W and W' we define the partial order $< . <'$ to be the relation $< \cup <' \cup \{w < w' \mid w \in W, w' \in W'\}$. Given two protocol specifications P_1, P_2, let $Names_1$, $Names_2$ their respective sets of principals. We can assume up to some renaming that for every principal A the associated partially ordered sets W_A^1 and W_A^2 in protocol P_1 and P_2 resp. are disjoint. Then we get a new protocol for the sequential composition $P_1.P_2$ by taking

$$\{R_\iota \Rightarrow S_\iota \mid \iota \in \mathcal{I}\}$$

where $\mathcal{I} = \{(A, i) \mid A \in Names_1 \cup Names_2 \text{ and } i \in W_A^1 \cup W_A^2\}$ For each principal A we associate now the partially ordered set $(W_A^1 \cup W_A^2, <_{W_A^1} . <_{W_A^2})$

We get similarly a protocol $P_1||P_2$ for parallel composition by taking for each principal A the partially ordered set $(W_A^1 \cup W_A^2, <_{W_A^1} \cup <_{W_A^2})$.

3 NP-hardness

We show now that the existence of an attack when the input are a protocol specification and initial knowledge of the intruder is NP-hard by reduction from 3-SAT. The proof is similar to the one given by [1] for their model.

- Propositional Variables $= x_1, .., x_n$.
- Instance of 3-SAT: $f(\overrightarrow{x}) = \bigwedge_I (x_{i,1}^{\varepsilon_{i,1}} \vee x_{i,2}^{\varepsilon_{i,2}} \vee x_{i,3}^{\varepsilon_{i,3}})$

where $\varepsilon_{i,j} \in \{0, 1\}$ and x^0 (resp.x^1) means x (resp.$\neg x$). Let us define

- $g(\varepsilon_{i,j}, x_{i,j}) = x_{i,j}$ if $\varepsilon_{i,j} = 0$
- $g(\varepsilon_{i,j}, x_{i,j}) = \{x_{i,j}\}_K$ if $\varepsilon_{i,j} = 1$
- $\forall i \in I$,
  $f_i(\overrightarrow{x}) = \langle g(\varepsilon_{i,1}, x_{i,1}), \langle g(\varepsilon_{i,2}, x_{i,2}), g(\varepsilon_{i,3}, x_{i,3})\rangle\rangle$

Let us introduce now the following protocol (where variables occurring in the description of step (U, j) should be considered as indexed by (U, j); the index is omitted for readability):

- Principal A:
  - $(A, 1) : \langle x_{1,1}, .., x_{n,3}\rangle \Rightarrow \{\langle f_1(\overrightarrow{x}), \langle f_2(\overrightarrow{x}), \langle ..., \langle f_n(\overrightarrow{x}), end\rangle\rangle\rangle\rangle\}_P$
- Principal B:
  - $(B, i) : \{\langle\langle \top, \langle x, y\rangle\rangle, z\rangle\}_P \Rightarrow \{z\}_P$
  - for $1 \leq i \leq |f(\overrightarrow{x})|$
- Principal B':
  - $(B', i) : \{\langle\langle \{\neg\top\}_K, \langle x, y\rangle\rangle, z\rangle\}_P \Rightarrow \{z\}_P$
  - for $1 \leq i \leq |f(\overrightarrow{x})|$
- Principals C, C', D, D': The same as B and B' for $\langle x, \langle \top, y\rangle\rangle$, $\langle x, \langle \{\neg\top\}_K, y\rangle\rangle$, $\langle x, \langle y, \top\rangle\rangle$, and $\langle x, \langle y, \{\neg\top\}_K\rangle\rangle$.
- Principal E: The rule $(E, 1) : \{end\}_P \Rightarrow Secret$

	Without Nonces	With Nonces
No bounds [11]	Undecidable	Undecidable
Infinite number of sessions, and Bounded messages [10]	DEXPTIME	Undecidable
Finite number of sessions, choice points, and Unbounded messages	NP-Complete	NP-Complete
Finite number of sessions, and Unbounded messages	NP-Complete	NP-Complete

Table 2. Complexity of known fragments

We take $S_0 = \{K^{-1}, \top, \neg\top\}$ for the initial intruder knowledge. Hence there is an attack on this protocol iff the message sent by principal A can be reduced to $\{end\}_P$ i.e. for all i, there exists j such that $g(\varepsilon_{i,j}, x_{i,j}) \in \{\top, \{\neg\top\}_K\}$, i.e. the intruder has given to A a term representing a solution of 3-SAT ($\{\neg\top\}_K$ represent $\top$). Hence the protocol admits an attack iff the corresponding 3-SAT problem has a solution. Moreover the reduction is obviously polynomial, hence the problem of finding an attack with bounded sessions is NP-hard.

From the results above we finally conclude with the main result:

Theorem 2 *Finding an attack for a protocol with a fixed number of sessions is an NP-complete problem.*

Conclusion

We have proved that when the number of sessions is fixed, in order to find an attack an intruder needs only to forge messages with polynomial size, when using a the dag-representation. We have also given an NP-procedure for finding an attack with a fixed number of sessions. Our formal model of protocols and attacks supports non-atomic symmetric keys. Several interesting variants of this model can be easily reduced to it.

If the intruder is allowed to generate any number of new datas, then we can prove with the same techniques that any attack he can launch can be also obtained by replacing in every message his freshly generated by his name (Charlie). The normal principals won't see any difference. Hence the intruder does not gain any strength when being able to create nonces, in the finite session case.

For instance we could consider that a principal is unable to recognize that a message supposed to be encrypted by some key K has really been constructed by an encryption with K, (see extension in Appendix 4.1). To obtain a protocol model where principals may recognize whether a real encryption has been performed one simply extend any cipher with a special fixed field.

We have considered that the intruder can eaves-drop, divert messages, and impersonate other principals. However we can model a more passive intruder, as described in Appendix 4.2, by ensuring that some messages cannot be modified (for instance when they are conveyed by a safe channel).

We have considered secrecy properties. Since correspondence attacks can also be expressed by an execution order and a polynomial number of $Forge$ constraints they can be detected in NP too.

Finally our procedure can also be adapted to protocols admitting choice points, where a different subprotocol can be executed by a user according to some received message. Protocols such as SSL admit choice points. The modification of our model is described in Appendix 4.3. The detection of an attack remains in NP. We can summarized the known results in the Table 2.

Directions for future works include broadening the scope of our approach to some cases where the number of sessions is unbounded or to commutativity of encryption operators.

4 Appendix

4.1 Extending the intruder model

The rules in Table 3 have to be added to the intruder rules in order to find attacks for protocols of the following type (we have omitted $init$ and end messages):

Intruder initial knowledge: $\{Secret\}_P$

$((A, 1), \{x\}_{K^{-1}} \Rightarrow x)$

$((A, 2), \{\{y\}_P\}_K \Rightarrow y)$

This protocol admits the following attack when the initial knowledge of intruder is $\{Charlie, \{Secret\}_P\}$:

$$\{Secret\}_P \to \{\{\{Secret\}_P\}_K\}_{K^{-1}} \\ \to \{\{Secret\}_P\}_K \to Secret$$

We can remark that such an attack cannot be found if the L_r rules are not included in the intruder rewrite system. However it is easy to prove that NP-completeness remains valid when these L_r and L_s rules are included. These new rules

Decomposition rules		Composition rules	
$L_s(\{\{a\}^s_b\}^s_b)$:	$\{\{a\}^s_b\}^s_b \to a, \{\{a\}^s_b\}^s_b$,	$L_r(\{\{a\}^s_b\}^s_b)$:	$a \to a, \{\{a\}^s_b\}^s_b$
$L_s(\{\{a\}^p_K\}^p_{K^{-1}})$:	$\{\{a\}^p_K\}^p_{K^{-1}} \to a, \{\{a\}^p_K\}^p_{K^{-1}}$	$L_r(\{\{a\}^p_K\}^p_{K^{-1}})$:	$a \to a, \{\{a\}^p_K\}^p_{K^{-1}}$

Table 3. Extension of the Intruder Model.

behave exactly in the same way as L_c and L_d, and we can restrict to consider derivations D such that $L_d(t') \in D$ or $L_s(t') \in D$ imply $L_c(t') \notin D$ and $L_r(t') \notin D$. (respectively L_c, L_r and L_d, L_s). This is stated by the following lemma:

Lemma 5 *If $t \in Forge(E)$, then there exists a derivation D from E with goal t, and verifying* Condition 1*: for all messages a, b ,*

1. *if $L_d(\{\{a\}_b\}_{b^{-1}}) \in D$ then $L_r(\{\{a\}_b\}_{b^{-1}}) \notin D$*
2. *if $L_c(\{a\}_b) \in D$ then $L_s(\{\{a\}_b\}_{b^{-1}}) \notin D$*
3. *if $L_s(\{\{a\}_b\}_{b^{-1}}) \in D$ then $L_c(\{\{a\}_b\}_{b^{-1}}) \notin D$*
4. *if $L_r(\{\{a\}_b\}_{b^{-1}}) \in D$ then $L_d(\{a\}_b) \notin D$*

Proof: Let D be one of the minimal derivations in length from E with goal t. Then let us build D' another derivation with the same initial and final set than D, verifying moreover *Condition 1*, and minimal in length among the derivations with these initial and final sets and verifying *Condition 1*. We reason by induction on the length of D. If $D = \emptyset$ we take $D' = \emptyset$. Otherwise $D = (E_0 \to_{L_1} .. \to_{L_n} E_n)$ and by induction hypothesis there exists $D'_1 = (E'_0 \to_{L'_1} .. \to_{L'_{n-1}} E'_{n-1})$ with $E_0 = E'_0$, $E_{n-1} = E'_{n-1}$, verifying *Condition 1*, and minimal in length. We show how to extend this to a derivation D' according to the last step of D:

1. if $L_n = L_d(\{\{\alpha\}_\beta\}\}_{\beta^{-1}})$ and there exists $i < n$ such that $L'_i = L_r(\{\{\alpha\}_\beta\}_{\beta^{-1}}) \in D'_1$, then let $L'_n = L_c(\{\alpha\}_\beta)$. L'_n can be applied since $\alpha \in E'_{i-1}$ and $\beta \in E_{i-1}$, and we have $L_s(\{\{\alpha\}_\beta\}_{\beta^{-1}}) \notin D'_1$ since otherwise D'_1 would not be minimal in length.
2. if $L_n = L_c(\{\alpha\}_\beta)$ and there exists $i < n$ such that $L'_i = L_s(\{\{\alpha\}_\beta\}_{\beta^{-1}}) \in D'_1$, then let $L'_n = L_d(\{\{\alpha\}_\beta\}_{\beta^{-1}})$. L'_n can be applied since $\{\{\alpha\}_\beta\}_{\beta^{-1}} \in E'_{i-1}$ and $\beta \in E'_{i-1}$, and we have $L_r(\{\{\alpha\}_\beta\}_{\beta^{-1}}) \notin D'_1$ by minimality of D'_1.
3. if $L_n = L_s(\{\{\alpha\}_\beta\}_{\beta^{-1}})$ and there exist $i < n$ such that $L'_i = L_c(\{\{\alpha\}_\beta\}_{\beta^{-1}}) \in D'_1$, then:
   - If $L_r(\{\alpha\}_\beta) \notin D'_1$, we take $L'_n = L_d(\{\alpha\}_\beta)$ since $\{\alpha\}_\beta$ and β are in E_{i-1}.
   - If $L_r(\{\alpha\}_\beta) \in D'_1$, then $\{\alpha\}_\beta = \{\{\alpha'\}_{\beta^{-1}}\}_\beta$ and we take $L'_n = L_c(\{\alpha'\}_{\beta^{-1}})$ since α' and β^{-1} are in E_{n-1}.
4. if $L_n = L_r(\{\{\alpha\}_\beta\}_{\beta^{-1}})$ and there exists $i < n$ such that $L'_i = L_d(\{\alpha\}_\beta) \in D'_1$, then:
   - If $L_s(\{\{\{\alpha\}_\beta\}_{\beta^{-1}}\}_\beta) \notin D'_1$, we take $L'_n = L_c(\{\{\alpha\}_\beta\}_{\beta^{-1}})$, since $\{\alpha\}_\beta$ and β are in E_{i-1}.
   - If $L_s(\{\{\{\alpha\}_\beta\}_{\beta^{-1}}\}_\beta) \in D'_1$, we take $L'_n = L_d(\{\{\{\alpha\}_\beta\}_{\beta^{-1}}\}_\beta)$.
5. otherwise in all remaining cases we take $R' = R$.

Then we can notice that $E'_n = E_n$ for $E'_{n-1} \to_{L'_n} E'_n$ and $D' = (E_0 \to_{L_0} .. \to_{L_n} E'_n)$ verify *Condition 1*. Hence, the set of derivations with the same initial and final sets than D and verifying *Condition 1* is not empty, and we can choose one of its elements minimal in length.

Hence from a derivation proving that $t \in Forge(E)$ we can build another one verifying moreover *Condition 1*. The minimal prefix of this derivation that contains t is a derivation of goal t satisfying *Condition 1*, and this proves the lemma. □

Now, we only need to update the $Deriv_t(E)$ definition:

Definition 7 *We denote $Deriv_t(E)$ a derivation of minimal length among the derivations from E with goal t and satisfying* Condition 1 *(chosen arbitrarily among the possible ones).*

The rest of the proof is almost identical except that L_c is replaced by L_c and L_r, and L_d is replaced by L_d and L_s. Note that this model can still be improved since here the L_s and L_r rules are only applied at the top of messages: we could also consider cases where they are applied everywhere in terms.

4.2 Limiting the intruder

In this section we show how to reduce to our model other models where the intruder is unable (for some messages) to eaves-drop, divert and modify, or impersonate.

To prevent the intruder from eaves-dropping between two steps (A, i) : ... $\Rightarrow M_1$ and (A, j) : $M_2 \Rightarrow$ ..., we introduce a new symmetric key P only known by A and B and never published: With the steps (A,i) : ... $\Rightarrow \{\langle A, M_1\rangle\}^s_P$ and (A,j) : $\{\langle A, M_2\rangle\}^s_P \Rightarrow$ ..., the intruder will never be able to intercept the message M_1 and send something else to B instead, even with the L_r rules.

To prevent the intruder from diverting and modifying the message between the two steps above, we can use a new private key $K_{A,i}$ instead of P, and the steps (A,i) : ... $\Rightarrow \{\langle A, M_1\rangle\}^p_{K_{A,i}}$ and (A,j) : $\{\langle A, M_2\rangle\}^p_{K_{A,i}} \Rightarrow$ .... This way, if the intruder know $K^{-1}_{A,i}$ then he knows M_1, but he cannot modify it, even with the L_r rules. And since $K_{A,i}$ is only used for this step, the intruder can't use any other stored message.

To prevent the intruder from impersonating a message M sent by the principal A to B, we assume that there exists a private key K only known by the principals and never published (it never appears in the content of a message), and we assume that K^{-1} is known by everybody. Hence this amounts to replace M by $\{\langle A, M\rangle\}^p_K$ in A and B's steps: The intruder and the principals can read M, but the intruder will never be able to build a message in the name of A, even with the L_r rules. (But he can use old A's messages if we used the same key K)

The new protocol has a linear size in the initial one, even in a dag representation.

4.3 Adding choice points

We extend the protocol model in order to allow for choice points. Typically the field of a message may contain information about the type of cryptography negotiated for the rest of the session. Hence the subsequent message exchanges may depend from the content of this field.

We shall consider protocol descriptions where some steps (A,i) may be composed by priority blocks:

$(A,(i,0))$: $R^0_i \Rightarrow S^0_i$
$(A,(i,1))$: $R^1_i \Rightarrow S^1_i$
...
$(A,(i,k))$: $R^k_i \Rightarrow S^k_i$

Two steps in the same priority block cannot be applied in the same execution, and $\forall j, j'$, $(A,(i,j)) <_{w_A} (A,(i+1,j'))$. This block construction is similar to a case structure in programming languages. A protocol execution with substitution σ must now also satisfy: For every priority block (indexed by i), we apply step $(A,(i,j))$ if $\forall j$, $\forall \delta$, $R^j_i\sigma \neq R^0_i\delta$ and ... and $R^j_i\sigma \neq R^{j-1}_i\delta$.

For technical reasons, we introduce for each variable x, a term $Charlie_x$ that does not appear anywhere in the protocol description, that is initially known only by the intruder and such that $|Charlie_x| = 0$. Then, an attack is now given by an order which is compatible with the given partial order and a substitution σ verifying the above steps conditions, the usual $R_i\sigma \in Forge(S_0\sigma,..,S_{i-1}\sigma)$ and $Secret \in Forge(S_0\sigma,..,S_n\sigma)$ conditions, and for all variables x, $Charlie_x$ may only appear in $\sigma(x)$. This last condition does not restrict the intruder since he is not forced to use $Charlie_x$.

First we can remark that all properties, lemmas, and proofs about derivations remain valid since we did not change the intruder rules. Therefore, the only proof to be adapted is the one of Lemma 4: We build a new substitution σ' from σ , and we must prove that it still satisfies the requirements for an attack. To do that, we assume first that $Charlie_x$ is used instead of $Charlie$ for $\sigma(x)$ in the proof of the lemma.

We have $R_i\sigma' \in Forge(S_0\sigma',..,S_{i-1}\sigma')$ and $Secret \in Forge(S_0\sigma',..,S_n\sigma')$. And for each attack step $\pi^{-1}(k) = (A,(i,j))$, we have $R^j_i\sigma \neq R^0_i\delta$ and ... and $R^j_i\sigma \neq R^{j-1}_i\delta$ by definition of σ. Let us show by contradiction that there is no δ such that $R^j_i\sigma' = R^{j'}_i\delta$ for some $j' < j$. Two cases are possible: If x appears in R^j_i, then $Charlie_x$ appears in $R^j_i\sigma'$ and in $R^{j'}_i\delta$. Therefore, we have $R^j_i\sigma = R^{j'}_i\delta'$ for δ' equal to δ with $Charlie_x$ replaced by $\sigma(x)$. But this is impossible by definition of σ. If x does not appear in R^j_i, then $R^j_i\sigma = R^j_i\sigma' = R^{j'}_i\delta$ and we also have a contradiction. This way, we have proved that σ' defines an attack with the same execution order and which is smaller than σ (since $|Charlie_x| = 0$. The lemma follows.

Since all bounds on derivations and attacks remain valid we only need to add to our insecurity detection procedure, an extra guessing step for the branches to be taken at choice points in order to derive an NP procedure for the more general case of protocols with choice.

4.4 Otway_Rees Protocol

```
Identifiers
 A, B, S : user;
 Kas,Kbs, Kab: symmetric_key;
 M,Na,Nb,X : number;
Knowledge
 A : B,S,Kas;
 B : S,Kbs;
 S : A,B,Kas,Kbs;

Messages
1. A -> B : M,A,B,{Na,M,A,B}Kas
2. B -> S : M,A,B,{Na,M,A,B}Kas
                 ,{Nb,M,A,B}Kbs
3. S -> B : M,{Na,Kab}Kas,{Nb,Kab}Kbs
```

$$
\begin{array}{llll}
\{ & ((A,1), & init & \Rightarrow \langle M,A,B\rangle, \{N_A, \langle M,A,B\rangle\}_{K_{AS}} \quad) \\
 & ((B,1), & \langle x_2,x_3,B\rangle, x_4 & \Rightarrow x_2,x_3,B,x_4,\{N_B,x_2,x_3,B\}_{K_{BS}} \quad) \\
 & ((S,1), & x_7,A,B,\{x_8,x_7,A,B\}_{K_{AS}}, \{x_9,x_7,A,B\}_{K_{BS}} & \Rightarrow x_7, \{x_8,K_{ab}\}_{K_{AS}}, \{x_9,K_{ab}\}_{K_{BS}} \quad) \\
 & ((B,2), & x_2,x_5,\{N_B,x_6\}_{K_{BS}} & \Rightarrow x_2,x_5 \quad) \\
 & ((A,2), & M,\{N_A,x_1\}_{K_{AS}} & \Rightarrow \{Secret\}_{x_1} \quad) \\
 & ((B,3), & \{Secret\}_{x_6} & \Rightarrow end \quad) \quad \}
\end{array}
$$

Figure 2. Otway_Rees protocol in our notation

```
4. B -> A : M,{Na,Kab}Kas
5. A -> B : {X}Kab
```

Let us write in Figure 2 the Otway_Rees protocol specification with our notation. For simplicity we write M, M', M'' for $\langle\langle M, M'\rangle, M''\rangle$

An execution can be obtained by taking the protocol steps in the given order and by taking the substitution is: $x_1 = K_{ab}$, $x_2 = M$, $x_3 = A$, $x_4 = \{\langle N_A, \langle M, \langle A, B\rangle\rangle\rangle\}_{K_{AS}}$, $x_5 = \{x_8, K_{ab}\}_{K_{AS}}$, $x_6 = K_{ab}$, $x_7 = M$, $x_8 = N_A$, $x_9 = N_B$. An attack can be performed on this protocol with initial intruder knowledge: $S_0 = \{Charlie\}$, using the substitution $[x_1 \leftarrow \langle M, \langle A, B\rangle\rangle]$ and the steps $(A,1)$, $(A,2)$ since:

$$
\begin{array}{c}
\langle M,A,B\rangle, \{N_A, M, A, B\}_{K_{AS}} \\
\rightarrow_{L_d}\rightarrow_{L_d}\rightarrow_{L_c} \; M, \{N_A, \langle M,A,B\rangle\}_{K_{AS}} \\
\text{and} \\
\{Secret\}_{\langle M,A,B\rangle}, \langle M,A,B\rangle, \{N_A, \langle M,A,B\rangle\}_{K_{AS}} \\
\rightarrow_{L_d}\rightarrow_{L_d} \; Secret
\end{array}
$$

4.5 Proof of Lemma 1

Given a set of terms E, a variable x and a message t, we want to show: $|E[x \leftarrow t]|_{DAG} \leq |E, t|_{DAG}$:

Let us first remark that we have $|t, t|_{DAG} = |t|_{DAG}$. We recall that $Sub(E')$ denotes the set of subterms of E'. We introduce a function $f : Sub(E[x \leftarrow t]) \longrightarrow Sub(E, t)$ and we show that f is one-to-one. Let us define f for $\alpha \in Sub(E[x \leftarrow t])$ by:

- $f(\alpha) = \alpha$ if $\alpha \in Sub(t)$.
- $f(\alpha) = \alpha'$ if $\alpha = \alpha'[x \leftarrow t]$ for some subterm α' of E

When several α' are possible in the definition above then we take one arbitrarily. Let us show that f is one-to-one. Consider $\alpha, \beta \in Sub(E[x \leftarrow t])$ with $\alpha \neq \beta$.

- If $\alpha, \beta \in Sub(t)$ then $f(\alpha) = \alpha$, $f(\beta) = \beta$, and $f(\alpha) \neq f(\beta)$.
- If $\alpha \in Sub(t)$ and $\beta = \beta'[x \leftarrow t]$, and $\beta' \in Sub(E)$, then $\alpha[x \leftarrow t] = \alpha \neq \beta'[x \leftarrow t]$, $\alpha \neq \beta'$ and so $f(\alpha) \neq f(\beta)$.
- If $\alpha = \alpha'[x \leftarrow t]$ and $\beta = \beta'[x \leftarrow t]$, with $\alpha', \beta' \in Sub(E)$, then $\alpha' \neq \beta'$ and $f(\alpha) \neq f(\beta)$.

This proves the property, since the DAG-size of a set of terms is equal to number of distinct subterms they contain. □

References

[1] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols, In *Proc. CONCUR 2000*, Springer Lecture Notes in Computer Science 1877.

[2] D. Basin. Lazy infinite-state analysis of security protocols. In *Secure Networking — CQRE [Secure] '99*, LNCS 1740, pages 30–42. Springer-Verlag, Berlin, 1999.

[3] D. Bolignano. Towards the formal verification of electronic commerce protocols. In *IEEE Computer Security Foundations Workshop*, pages 133–146. IEEE Computer Society, 1997.

[4] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov. A meta-notation for protocol analysis. In P. Syverson, editor *12th IEEE Computer Security Foundations Workshop* IEEE Computer Society Press, 1999.

[5] E.M. Clarke, and S. Jha, W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET), 1998.

[6] G. Denker, J. Meseguer, and C. Talcott. Protocol specification and analysis in Maude. In Heintze, N. and Wing, J., editors, Proc. of Workshop on Formal Methods and Security Protocols, 25 June 1998, Indianapolis, Indiana,

[7] G. Denker and J. Millen. CAPSL Integrated Protocol Environment. In DARPA Information and Survivability Conference and Exposition (DISCEX'00), Hilton Head, South Carolina, January 25-27, 2000, pp. 207-221, IEEE Computer Society Press

[8] G. Denker, J. Millen, F. Küster Filipe and A. Grau. Optimizing Protocol Rewrite Rules of CIL Specifications. In 13th IEEE Computer Security Foundations Workshop, July 3-5, 2000, Cambridge, England

[9] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29:198–208, 1983. Also STAN-CS-81-854, May 1981, Stanford U.

[10] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov. Undecidability of Bounded Security Protocols Workshop on Formal Methods and Security Protocols July 5, 1999, Trento, Italy (part of FLOC'99)

[11] S. Even, O. Goldreich. On the security of multi-party ping pong protocols. Technical Report 285, Israel Institute of Technology, 1983.

[12] Jean Goubault-Larrecq A Method for Automatic Cryptographic Protocol Verification (Extended Abstract). In *Proc. Workshop on Formal Methods in Parallel Programming, Theory and Applications (FMPPTA'2000)*, LNCS 1800, pages 977–984, Springer, 2000.

[13] J. Heather and S. Schneider. Towards automatic verification of authentication protocols on an unbounded network *13th IEEE Computer Security Foundations Workshop*, July 2000.

[14] A. Huima. Efficient infinite-state ananysis of security protocols. LICS'99, Workshop on Formal Methods and Security Protocols, 1999.

[15] F. Jacquemard, M. Rusinowitch and L. Vigneron. Compiling and Verifying Security Protocols. Logic for Programming and Automated Reasoning. St Gilles, Reunion Island. Springer Verlag, 2000. LNCS. vol 1955. 30 p. M. Parigot and A. Voronkov editors.

[16] G. Lowe. Casper: a compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1):53–84, 1998.

[17] G. Lowe. Towards a completeness result for model checking of security protocols. In *11th IEEE Computer Security Foundations Workshop*, pages 96–105. IEEE Computer Society, 1998.

[18] C. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1):5–36, 1992.

[19] C. Meadows. The NRL protocol analyzer: an overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[20] F. Martinelli Languages for the description and the analysis of authentication protocols. In Proc. of ICTCS'98. World Scientific Press.

[21] J. Millen. CAPSL: Common Authentication Protocol Specification Language. Technical Report MP 97B48, The MITRE Corporation, 1997.

[22] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murϕ. In *IEEE Symposium on Security and Privacy*, pages 141–154. IEEE Computer Society, 1997.

[23] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[24] A. W. Roscoe. Modelling and verifying key exchange protocols using CSP and FDR. In *8th IEEE Computer Security Foundations Workshop*, pages 98–107. IEEE Computer Society, 1995.

[25] S Schneider Verifying Authentication Protocols with CSP. 10th IEEE Computer Security Foundations Workshop, 1997.