INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An NP Decision Procedure for Protocol Insecurity with XOR

Yannick Chevalier — Ralf Küsters — Michaël Rusinowitch — Mathieu Turuani

N° 4697

January 2003

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4697--FR+ENG

An NP Decision Procedure for Protocol Insecurity with XOR

Yannick Chevalier , Ralf Küsters * , Michaël Rusinowitch , Mathieu Turuani

Thème 2 — Génie logiciel
et calcul symbolique
Projet Cassis

Rapport de recherche n° 4697 — January 2003 — 22 pages

Abstract: We provide a method for deciding the insecurity of cryptographic protocols in presence of the standard Dolev-Yao intruder (with a finite number of sessions) extended with so-called oracle rules, i.e., deduction rules that satisfy certain conditions. As an instance of this general framework, we obtain that protocol insecurity is in NP for an intruder that can exploit the properties of the XOR operator. This operator is frequently used in cryptographic protocols but cannot be handled in most protocol models. We also apply our framework to an intruder that exploits properties of certain encryption modes such as cipher block chaining (CBC).

Key-words: Verification, Rewriting, Reasoning about Security, Attack, Protocol, Theorem Proving, Logic and Complexity

* Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

L'insécurité des protocoles cryptographiques avec XOR est NP-complet

Résumé : Nous présentons une méthode pour rechercher les attaques sur les protocoles cryptographiques (avec un nombre fini de sessions) en présence d'un intrus suivant le modèle de Dolev-Yao étendu par des règles d'oracle, c.à.d des règles de déduction vérifiant certaines conditions. En particulier, cette méthode générale assure que la recherche d'une attaque en présence d'un intrus capable d'utiliser les propriétées de l'opérateur XOR est NP-complet. Cet opérateur est fréquemment utilisé dans les protocoles cryptographiques, mais peu d'analyses peuvent le prendre en compte. Nous utilisons également cette méthode générale avec un intrus capable d'exploiter certaines propriétés de certains modes d'encryption comme le chiffrement avec chainage des blocs (CBC).

Mots-clés : Vérification, Réécriture, Analyse de Sécurité, Attaque, Protocole, Preuve Automatique, Logique et Complexité

1 Introduction

Cryptographic protocols have been designed for handling secure electronic communications. Verification tools based on formal methods (e.g. model checking) have been quite successful in discovering new flaws in well-known security protocols [13, 17, 23, 3, 6].

While most formal analysis of security protocols abstracts from low-level properties, i.e., certain algebraic properties of encryption, such as the multiplicativity of RSA or the properties induced by chaining methods for block ciphers, many real attacks and protocol weaknesses rely on these properties. A typical example was provided by Ryan and Schneider [22] where they give a simple attack on Bull's recursive authentication protocol: the protocol is used to distribute a connected chain of keys linking all the nodes from originator to the server, but if one key is compromise the others can be comprised too thanks to the property of XOR. Conversely, if XOR is considered as a free operator then, as shown by L. Paulson using the Isabelle prover [19], the key distribution is secure.

Recently, several procedures have been proposed to decide insecurity of cryptographic protocols w.r.t. a finite number of protocol sessions [2, 5, 11, 21, 16, 12]. Also some special cases for an unbounded number of sessions have been studied [9, 4, 10, 8, 1]. All these results assume encryption to be perfect (*perfect encryption assumption*): One needs a decryption key to extract the plaintext from the ciphertext, and also, a ciphertext can be generated only with the appropriate key and message (no collision). Only very few works on formal analysis have relaxed this assumption. In [15], a unification algorithm is designed for handling properties of Diffie-Hellman cryptographic systems.

In this paper, we generalize the decidability result of [21], stating that insecurity for finitely many protocol sessions is in NP, to the case where messages may contain the XOR operator and where the Dolev-Yao intruder is extended by the ability to compose messages with the XOR operator. More precisely, we give a linear bound on the size of messages exchanged in minimal attacks and present an NP procedure for deciding insecurity with XOR. This extension is non-trivial due to the complex interaction of the XOR properties and the standard Dolev Yao intruder rules. The technical problems raised by the equational laws are somewhat related to those encountered in semantic unification.

To prove our result, we have extended the Dolev Yao intruder with so-called oracle rules, i.e., deduction rules that satisfy certain conditions. In this general framework we show that insecurity is decidable in NP. Now, the results for XOR are obtained by proving that the XOR rules satisfy the conditions on oracle rules.

Our framework is general enough to also handle other algebraic properties. More specifically, we show that the Dolev Yao intruder equipped with the ability to exploit prefix properties of encryption algorithms based on cipher-block-chaining (CBC), falls into our framework as well.

To our knowledge, the decidability results presented here are the first in formal analysis of security protocols that go beyond the perfect encryption hypothesis.[1]

Structure of the paper. In the following section, we provide an example illustrating the role of XOR in attacks. We then, in Section 3, introduce our protocol and intruder model. In particular, this section contains the definition of the oracle rules. The decidability result for the general framework is presented in Section 4, including the description of the NP decision algorithm. In Section 5, we show how to bound the size of minimal attacks. Then, in Section 6 XOR rules and prefix rules are introduced and it is shown that these rules are oracle rules, which implies the mentioned complexity results. In the appendix, the use of prefix rules is illustrated. Also, the proofs omitted in the paper are provided.

2 A Motivating Example

We illustrate that when taking the algebraic properties of XOR into account, new attacks can occur. As an example, we use a variant of the Needham-Schroeder-Lowe Protocol [14], i.e., the public-key Needham-Schroeder

[1] Recently, H. Comon and V. Shmatikov have announced related results.

Procotol with Lowe's fix, where in some place, instead of concatenation XOR is used. Using common notation, the protocol is given as follows:

$$\begin{array}{l} 1.\ A \to B:\ \{N_A, A\}^p_{K_B} \\ 2.\ B \to A:\ \{N_B, \mathrm{XOR}(N_A, B)\}^p_{K_A} \\ 3.\ A \to B:\ \{N_B\}^p_{K_B} \end{array}$$

If XOR is interpreted as free symbol, such as pairing, then according to [14] this protocol is secure. In particular, the intruder is not able to get hold of N_B. However, if the algebraic properties of XOR are taken into account, the following attack is possible, which is a variant of the original attack on the Needham-Schroeder Protocol and which allows the intruder I to obtain N_B. In this attack, two sessions run interleaved where the steps of the second session are marked with $'$. In the first session, A talks to the intruder I, and in the second session I, purporting to be A, talks to B. We emphasize that in this attack I generates new messages by applying the XOR operator and uses that $\mathrm{XOR}(N_A, B, I, B) =_{\mathrm{XOR}} \mathrm{XOR}(N_A, I)$.

$$\begin{array}{lll} 1.\ A \to I & : & \{N_A, A\}^p_{K_I} \\ 1'.\ I(A) \to B & : & \{\mathrm{XOR}(N_A, B, I), A\}^p_{K_B} \\ 2'.\ B \to I(A) & : & \{N_B, \mathrm{XOR}(N_A, B, I, B)\}^p_{K_A} \\ 2.\ I \to A & : & \{N_B, \mathrm{XOR}(N_A, B, I, B)\}^p_{K_A} \\ 3.\ A \to I & : & \{N_B\}^p_{K_I} \end{array}$$

3 The Protocol and Intruder Model

The protocol and intruder model we describe here extend standard models for the (automatic) analysis of security protocols [2, 10, 21, 16] in two respects. First, messages can be build using the XOR operator, which is not allowed in most other protocol models. Second, in addition to the standard Dolev Yao rewrite rules, the intruder is equipped with the mentioned oracle rules. In what follows, we provide a formal definition of our model by defining terms, messages, protocols, the intruder, and attacks.

3.1 Terms and Messages

First, recall that a finite multiset over a set S is a function M from S to $\mathbb{N}$ with finite domain. We use the common set notation to define multisets. For example, $\{a, a, a, b\}$ denotes the multiset M with $M(a) = 3$, $M(b) = 1$, and $M(x) = 0$ for every $x \notin \{a, b\}$.

Terms are defined according to the following grammar:

$$term ::= Atoms \mid Var \mid \langle term, term\rangle \mid \{term\}^s_{term} \mid \{term\}^p_{Keys} \mid \mathrm{XOR}(M)$$

where $Atoms$ is a finite set of constants (*atomic messages*), containing principal names, nonces, keys, and the constants 0 and $Secret$; $Keys$ is a subset of $Atoms$ denoting the set of public and private keys; Var is a finite set of variables; and M is a non-empty finite multiset of terms. We assume that there is a bijection $\cdot^{-1}$ on $Keys$ which maps every public (private) key k to its corresponding private (public) key k^{-1}. The binary symbol $\langle\cdot,\cdot\rangle$ is called *pairing*, the binary symbol $\{\cdot\}^s_\cdot$ is called *symmetric encryption*, the binary symbol $\{\cdot\}^p_\cdot$ is *public key encryption*. Note that a symmetric key can be any term and that for public key encryption only atomic keys (namely, public and private keys from $Keys$) can be used. A term with head XOR is called *non standard* and otherwise it is called *standard*. Because of the algebraic properties of XOR (see below), it is convenient to define the XOR operator as done above, instead of defining it as a binary operator. We abbreviate $\mathrm{XOR}(\{t_1, \ldots, t_n\})$ by $\mathrm{XOR}(t_1, \ldots, t_n)$.

Variables are denoted by x, y, terms are denoted by s, t, u, v, and finite sets of terms are written $E, F, ...$, and decorations thereof, respectively. We abbreviate $E \cup F$ by E, F, the union $E \cup \{t\}$ by E, t, and $E \setminus \{t\}$ by $E \setminus t$. The same abbreviations are used for multisets.

For a term t and a set of terms E, $Var(t)$ and $Var(E)$ denote the set of variables occurring in t and E, respectively.

A *ground term* (also called *message*) is a term without variables. A *(ground) substitution* is a mapping from Var to the set of (ground) terms. The application of a substitution σ to a term t (a set of terms E) is written $t\sigma$ ($E\sigma$), and is defined as usual.

Given two terms u, v, the *replacement* of u by v, denoted by $[u \leftarrow v]$, maps every term t to the term $t[u \leftarrow v]$ which is obtained by replacing all occurrences of u in t by v. Note that the result of such a replacement is uniquely determined. We can compose a substitution σ with a replacement δ: the substitution $\sigma\delta$ maps every $x \in Var$ to $\sigma(x)\delta$.

The multiset of *factors* of a term t, denoted by $Factor(t)$, is recursively defined: If $t = \text{XOR}(M)$, then $Factor(t) = \uplus_{t' \in M} Factor(t')$, and otherwise, if t is standard, $Factor(t) = \{t\}$, where $\uplus$ is the union of multisets. Note that $Factor(t)$ only contains standard terms. For example, with $a, b, c \in Atoms$, $\{c, c, \langle \text{XOR}(a, b), c\rangle\} = Factor(\text{XOR}(c, \langle \text{XOR}(a, b), c\rangle, c))$.

The set of *subterms* of a term t, denoted by $Sub(t)$, is defined as follows:

- If $t \in Atoms$ or $t \in Var$, then $Sub(t) = \{t\}$.
- If $t = \langle u, v\rangle$, $\{u\}^s_v$, or $\{u\}^p_v$, then $Sub(t) = \{t\} \cup Sub(u) \cup Sub(v)$.
- If t is non standard, then $Sub(t) = \{t\} \cup \bigcup_{u \in \mathcal{T}} Sub(u)$ with $\mathcal{T}$ the multiset of factors of t.

We define $Sub(E) = \bigcup_{t \in E} Sub(t)$. Note that $\text{XOR}(a, b)$ is not a subterm of $\text{XOR}(\text{XOR}(a, b), c)$.

The *subterm ordering* $\leq$ is defined as usual: $t \leq t'$ if $t \in Sub(t') \setminus t'$.

We define the size of a term and a set of terms basically as the size of the representation as a dag. That is, the *size* $|t|$ ($|E|$) of a term t (a set of terms E) is $|Sub(t) \setminus \{0\}|$ ($|Sub(E) \setminus \{0\}|$). Note that 0 has size 0. In what follows, if $|\cdot|$ is applied to sets of terms, it will always denote the size of the set as just defined rather than the cardinality of the set, unless otherwise stated.

The XOR operator is considered to be commutative, associative, nilpotent, and 0 is the unit element. According to these properties, the normal form of a term is defined as the result of the *normalization function* $\ulcorner \_ \urcorner : term \rightarrow term$ defined recursively by:

- For an atom or a variable a, $\ulcorner a \urcorner := a$,
- For terms u and v, $\ulcorner \langle u, v\rangle \urcorner := \langle \ulcorner u \urcorner, \ulcorner v \urcorner \rangle$, $\ulcorner \{u\}^s_v \urcorner = \{\ulcorner u \urcorner\}^s_{\ulcorner v \urcorner}$, and $\ulcorner \{u\}^p_v \urcorner = \{\ulcorner u \urcorner\}^p_v$.
- For a non-standard term t, define M to be the multiset of factors of t in normalized form, i.e.,

$$M(t') := \sum_{t'', \ulcorner t'' \urcorner = t'} Factor(t)(t'')$$

  for every term t'. (Recall that $Factor(t)$ is a multiset.) Now, let $M'(t') := M(t') \bmod 2$ for every term t' with $M(t') \neq 0$, i.e., M' contains a term if and only if this term occurs in M and the number of occurrences in M is odd. Now, if $M'(t') = 0$ for every $t' \neq 0$, then $\ulcorner t \urcorner := 0$. Otherwise, define $M''(t') := M'(t')$ for every $t \neq 0$, and $M''(0) := 0$. If $M''(t') \neq 0$ for exactly one t', then $\ulcorner t \urcorner = t'$. Otherwise, $\ulcorner t \urcorner := \text{XOR}(M'')$.

The normalization function extends to sets, multisets of terms, and substitutions in the obvious way. A term t is *normalized* if $\ulcorner t \urcorner = t$. In the same way normalized sets, multisets of terms, and subsitutions are defined. Two terms t and t' are *equivalent* (modulo XOR) if $\ulcorner t \urcorner = \ulcorner t' \urcorner$. In this case, we write $t =_{\text{XOR}} t'$.

To illustrate the normalization function, we provide some examples: If $a, b, c, d \in Atoms$, then

$$\begin{aligned}
\ulcorner \text{XOR}(\text{XOR}(a, b, d), \text{XOR}(c, d)) \urcorner &= \text{XOR}(a, b, c) \\
\ulcorner \langle \text{XOR}(0, a, a, b, c), \text{XOR}(a, \text{XOR}(a, c))\rangle \urcorner &= \langle \text{XOR}(b, c), c\rangle \\
\ulcorner \text{XOR}(a, \langle \text{XOR}(b), a\rangle, c) \urcorner &= \text{XOR}(a, \langle b, a\rangle, c) \\
\text{However, } \ulcorner \text{XOR}(\langle a, b\rangle, \langle a, c\rangle) \urcorner &\neq \langle 0, \text{XOR}(b, c)\rangle .
\end{aligned}$$

One easily shows:

Lemma 1 *For every $n \geq 0$, term t, and substitution σ:*

1. $|\ulcorner t \urcorner| \leq |t|$, *and*
2. $\ulcorner t\sigma \urcorner = \ulcorner\ulcorner t \urcorner \sigma \urcorner = \ulcorner t \ulcorner \sigma \urcorner\urcorner = \ulcorner\ulcorner t \urcorner \ulcorner \sigma \urcorner\urcorner$.

We finally remark:

Remark 1 *For every normalized term t with $|t| \leq n$, the number of arguments of* XOR *operators occurring in t is bounded by n. Therefore, representing t (as a DAG) needs space polynomially bounded in n.*

3.2 Protocols

The following definition is explained below.

Definition 1 *A* protocol rule *is of the form $R \Rightarrow S$ where R and S are terms.*

A protocol *P is a tuple $(\{R_\iota \Rightarrow S_\iota, \iota \in \mathcal{I}\}, <_\mathcal{I}, \mathcal{S})$ where $\mathcal{S}$ is a finite normalized set of messages with $0 \in \mathcal{S}$, the* initial intruder knowledge, *$\mathcal{I}$ is a finite (index) set, $<_\mathcal{I}$ is a partial ordering on $\mathcal{I}$, and $R_\iota \Rightarrow S_\iota$, for every $\iota \in \mathcal{I}$, is a protocol rule such that*

1. *the terms R_ι and S_ι are normalized;*
2. *for all $x \in Var(S_\iota)$, there exists $\iota' \leq \iota$ such that $x \in Var(R_{\iota'})$;*
3. *for any subterm $\text{XOR}(t_1, \ldots, t_n)$ of R_ι, there exists $j \in \{1, \ldots, n\}$ such that $Var(t_i) \subseteq \cup_{\iota' <_\mathcal{I} \iota} Var(R_{\iota'})$ for every $i \in \{1, \ldots, n\} \setminus \{j\}$. (Note that since R_ι is normalized, the t_i's are standard terms.)*

A bijective mapping $\pi : \mathcal{I}' \to \{1, \ldots, |\mathcal{I}'|\}$, is called execution order *for P if $\mathcal{I}' \subseteq \mathcal{I}$, for all i, j: $i <_\mathcal{I} j$ and $\pi(j)$ is defined implies that $\pi(i)$ is defined, and for all i, j: $i <_\mathcal{I} j$ implies $\pi(i) < \pi(j)$. We define the size of π as $|\mathcal{I}'|$.*

Given a protocol P, in the following we will assume that *Atoms* is the set of constants occurring in P. We define $|P| := |Sub(\mathcal{S} \cup \bigcup_{\iota \in \mathcal{I}}(R_\iota \cup S_\iota))|$ to be the *size of P*. We define $Var = Var(P)$ to be the set of variables occurring in P.

Intuitively, when executing a rule $R_\iota \Rightarrow S_\iota$ and on receiving a (normalized) message m in a protocol run, it is first checked whether m and R_ι match, i.e., whether there exists a ground substitution σ such that $m =_{\text{XOR}} R_\iota\sigma$. If so, $\ulcorner S_\iota\sigma \urcorner$ is returned as output. We always assume that the messages exchanged between principals (and the intruder) are normalized — therefore, m is assumed to be normalized and the output of the above rule is not $S_\iota\sigma$ but $\ulcorner S_\iota\sigma \urcorner$. This is because principals and the intruder cannot distinguish between equivalent terms, and therefore, they may only work on normalized terms (representing the corresponding equivalence class of terms). Finally, we note that since the different protocol rules may share variables, some of the variables in R_ι and S_ι may be bounded already by substitutions obtained from applications of previous protocol rules. We are not actually interested in a normal execution of a protocol but rather in attacks on a protocol. This is the reason why the definition of a protocol contains the initial intruder knowledge. Attacks are formally defined in Section 3.3.

Condition 1. , in the above definition is not a real restriction since due to Lemma 1, the transformation performed by a protocol rule and its normalized variant coincide. Condition 2. guarantees that when with S_ι an output is produced, all variables in S_ι are "bounded" already. Otherwise, the output of a protocol rule would be arbitrary, since unbounded variables could be mapped to any message. Condition 3. guarantees that the bounding of variables is deterministic. For example, if the protocol rule $\text{XOR}(x, y) \Rightarrow \langle x, y \rangle$ does not have predecessors according to $<_\mathcal{I}$, and thus, x and y are not bounded, then this rule violates Condition 3: On receiving $\text{XOR}(a, b, c)$, for instance, different substitutions are possible, including $\{x \mapsto \text{XOR}(a, b), y \mapsto c\}$, $\{x \mapsto \text{XOR}(b, d), y \mapsto \text{XOR}(a, c, d)\}$, etc. In other words, a principal must guess a substitution. With Condition 3. we avoid this.

The protocol informally described in Section 2 can formally be stated as follows: agent a plays role A and agent b role B; $Atoms = \{0, a, b, na, nb, ka, kb, I, ki, ki'\}$; we define $\mathcal{I} = \{(a,1),(a,2),(b,1)\}$ and $<_{\mathcal{I}} := \{((a,1),(a,2))\}$; the initial knowledge of the intruder is $\mathcal{S} = \{0, I, ki, ki', ka, kb\}$, and the protocol rules are:

$$\begin{array}{lccc} (a,1): & 0 & \Rightarrow & \{\langle na, a\rangle\}^p_{kb} \\ (b,1): & \{\langle x_{na}, a\rangle\}^p_{kb} & \Rightarrow & \{\langle nb, \mathrm{XOR}(x_{na}, b)\rangle\}^p_{ka} \\ (a,2): & \{\langle x_{nb}, \mathrm{XOR}(na, b)\rangle\}^p_{ka} & \Rightarrow & \{x_{nb}\}^p_{kb} \end{array}$$

3.3 The Intruder Model and Attacks

Our intruder model follows the Dolev-Yao intruder [9]. That is, the intruder has complete control over the network and he can derive new messages from his initial knowledge and the messages received from honest principals during protocol runs. To derive a new message, the intruder can compose and decompose, encrypt and decrypt messages, in case he knows the key. What distinguishes the intruder we consider here from the standard Dolev-Yao intruder, is that we will equip the intruder with guess rules, which provide him with additional capabilities of deriving messages. In Section 3.4, we consider classes of guess rules with certain properties, so-called oracle rules. As mentioned, in Section 6 we will look at two different instances of these oracle rules, namely XOR and prefix rules.

The intruder derives new messages from a given (finite) set of message by applying intruder rules. An *intruder rule* (or *t-rule*) L is of the form $M \to t$, where M is a finite multiset of messages and t is a message. Given a finite set E of messages, the rule L *can be applied* to E if M is a subset of E, in the sense that if $M(t') \neq 0$, then $t' \in E$ for every message t'. We define the *step relation* $\to_L$ induced by L as a binary relation on (finite) sets of messages. For every finite set of messages E: $E \to_L E, t$ (recall that E, t stands $E \cup \{t\}$) if L is a t-rule and L can be applied to E. If $\mathcal{L}$ denotes a (finite or infinite) set of intruder rules, then $\to_{\mathcal{L}}$ denotes the union $\bigcup_{L\in\mathcal{L}} \to_L$ of the step relations $\to_L$ with $L \in \mathcal{L}$. With $\to^*_{\mathcal{L}}$ we denote the reflexive and transitive closure of $\to_{\mathcal{L}}$.

The set of intruder rules we consider in this paper is listed in Table 1. In this table, a, b denote (arbitrary) messages, K is an element of $Keys$, and E is a finite set of messages (considered as multiset).

We emphasize that the notion *intruder rule* will always refer to the rules listed in Table 1. For now, there may be any set of guess rules of the kind shown in Table 1, later we will consider certain classes of guess rules, namely oracle rules.

The intruder rules are denoted as shown in Table 1. With $L_{od}(a)$ and $L_{oc}(a)$ we denote (finite or infinite) sets of guess rules. For uniformity, we therefore consider $L_{p1}(\langle a, b\rangle), \ldots, L_{sd}(\{a\}^s_b)$ and $L_c(\langle a, b\rangle), \ldots, L_c(\{a\}^s_b)$ as singletons. Note that, even if there are no guess rules, the number of decomposition and composition rules is always infinite since there are infinitely many messages a, b.

We further group the intruder rules as follows. In the following unions, a ranges over all messages.

- $L_d(t) := L_{p1}(t) \cup L_{p2}(t) \cup L_{ad}(t) \cup L_{sd}(t)$ for every message t. In case, for instance, $L_{p1}(t)$ is not defined, i.e., the head symbol of t is not a pair, then $L_{p1}(t) = \emptyset$; analogously for the other rule sets,
- $L_d := \bigcup_a L_d(a)$, $L_c := \bigcup_a L_c(a)$, $L_{od} := \bigcup_a L_{od}(a)$, $L_{oc} := \bigcup_a L_{oc}(a)$,
- $L_o(t) := L_{oc}(t) \cup L_{od}(t)$, $L_o := L_{oc} \cup L_{od}$,
- $\mathcal{L}_d(t)$ is the set of all decomposition t-rules in Table 1, i.e., all t-rule in the left column of the table,
- $\mathcal{L}_c(t)$ is the set of all composition t-rules in Table 1.
- $\mathcal{L}_d := \bigcup_a \mathcal{L}_d(a)$, $\mathcal{L}_c := \bigcup_a \mathcal{L}_c(a)$, and $\mathcal{L} := \mathcal{L}_d \cup \mathcal{L}_c$.

Note that $\mathcal{L}$ denotes the (infinite) set of all intruder rules we consider here. The set of messages the intruder can derive from a (finite) set E of messages is:

$$forge(E) := \bigcup \{E' \mid E \to^*_{\mathcal{L}} E'\}.$$

From the definition of intruder rules in Table 1 it immediately follows:

	Decomposition rules	Composition rules
Pair	$L_{p1}(\langle a,b\rangle)$: $\langle a,b\rangle \to a$ $L_{p2}(\langle a,b\rangle)$: $\langle a,b\rangle \to b$	$L_c(\langle a,b\rangle)$: $a,b \to \langle a,b\rangle$
Asymmetric	$L_{ad}(\{a\}^p_K)$: $\{a\}^p_K, K^{-1} \to a$	$L_c(\{a\}^p_K)$: $a,K \to \{a\}^p_K$
Symmetric	$L_{sd}(\{a\}^s_b)$: $\{a\}^s_b, b \to a$	$L_c(\{a\}^s_b)$: $a,b \to \{a\}^s_b$
Guess	$L_{od}(a)$: $E \to a$ with a subterm of E and E normalized.	$L_{oc}(a)$: $E \to a$ with E, a normalized.

Table 1: Intruder Rules

Lemma 2 *If E is a normalized set of messages, then $forge(E)$ is normalized.*

The lemma says that if an intruder only sees normalized messages, then he only creates normalized messages. Intruders should be modeled in such a way that they cannot distinguish between equivalent messages since if one thinks of, for instance, the message $\text{XOR}(a,a,b)$, which is equivalent to b, as a bit string obtained by "XORing" the bit strings a, a, and b, then this bit string is simply b. Therefore, in what follows we always assume that the intruder's knowledge consists of a set of normalized messages, where every single normalized message in this set can be seen as a representative of its equivalence class.

We are now prepared to define attacks. In an attack on a protocol P, the intruder (nondeterministically) chooses some execution order for P and then tries to produce input messages for the protocol rules. These input messages are derived from the intruder's initial knowledge and the output messages produced by executing the protocol rules. The aim of the intruder is to derive the message $Secret$. If different sessions of a protocol running interleaved shall be analysed, then these sessions must be encoded into the protocol P. This is the standard approach when protocols are analysed w.r.t. a bounded number of sessions, see, for instance, [21].

Definition 2 *Let $P = (\{R'_\iota \Rightarrow S'_\iota \mid \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol. Then an* attack *on P is a tuple (π,σ) where π is an execution order on P and σ is a normalized ground substitution of the variables occurring in P such that $\ulcorner R_i\sigma \urcorner \in forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma\urcorner)$ for every $i \in \{1,\dots,k\}$ where k is the size of π, $R_i := R'_{\pi^{-1}(i)}$, and $S_i := S'_{\pi^{-1}(i)}$ and such that $Secret \in forge(\ulcorner S_0, S_1\sigma, ..., S_k\sigma\urcorner)$.*

Due to Lemma 1, it does not matter whether, in the above definition, σ is normalized or not. Also note that Lemma 2 implies: $\ulcorner forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma\urcorner)\urcorner = forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma\urcorner)$.

The decision problem we are interested in is the following set of protocols:

$$\text{INSECURE} := \{P \mid \text{ there exists an attack on } P\}.$$

3.4 Oracle Rules

Oracle rules are guess rules which satisfy certain conditions. To define these rules, we first need some new notions.

A *derivation* D of length n, $n \geq 0$, is a sequence of steps of the form $E \to_{L_1} E, t_1 \to_{L_2} \cdots \to_{L_n} E, t_1, \dots, t_n$ with a finite set of messages E, messages $t_1,\dots,t_n$, intruder rules $L_i \in \mathcal{L}$, such that $E, t_1, \dots, t_{i-1} \to_{L_i} E, t_1, \dots, t_i$ and $t_i \notin E \cup \{t_1,\dots,t_{i-1}\}$, for every $i \in \{1,\dots,n\}$. The rule L_i is called the *ith rule* in D and the step $E, t_1, \dots, t_{i-1} \to_{L_i} E, t_1, \dots, t_i$ is called the *ith step* in D. We write $L \in D$ to say that $L \in \{L_1,\dots,L_n\}$. If S is a set of intruder rules, then we write $S \notin D$, to say $S \cap \{L_1,\dots,L_n\} = \emptyset$. The message t_n is called the *goal* of D.

We also need *well formed* derivations which are derivations where every message generated by an intermediate step in the derivation either occurs in the goal or in the initial set of messages.

Definition 3 *Let $D = E \to_{L_1} \dots \to_{L_n} E'$ be a derivation with goal t. Then, D is* well formed *if for every $L \in D$ and any t': $L \in \mathcal{L}_c(t')$ implies $t' \in Sub(E,t)$, and $L \in \mathcal{L}_d(t')$ implies $t' \in Sub(E)$.*

We can now define oracle rules. Condition 1. in the following definition will allow us to bound the length of derivations. The remaining conditions are later used to bound the size of the substitution σ of an attack: Condition 3. restricts the kind of output an L_{oc} rule can produce. Condition 2. and 4. will allow us to replace a subterm u in σ, composed by the intruder, by a smaller message.

Definition 4 *Let $L_o = L_{oc} \cup L_{od}$ be a (finite or infinite) set of guess rules, where L_{oc} and L_{od} denote disjoint sets of composition and decomposition guess rules, respectively. Then, L_o is a* set of oracle rules *(w.r.t. $L_c \cup L_d$ as defined above) iff:*

1. *For every message t, if $t \in forge(E)$, then there exists a well formed derivation from E with goal t.*
2. *If $F \rightarrow_{L_{oc}(t)} F, t$ and $F, t \rightarrow_{L_d(t)} F, t, a$, then there exists a derivation D from F with goal a such that $L_d(t) \notin D$.*
3. *For every rule $F \rightarrow s \in L_{oc}(s)$, every proper subterm of s is a subterm of F.*
4. *For every non atomic message u, there exists a normalized message $\epsilon(u)$ with $|\epsilon(u)| < |\ulcorner u \urcorner|$ such that: For every finite set F of messages with $0 \in F$, if $F \setminus u \rightarrow_{\mathcal{L}_c(u)} F$, i.e., u can be composed from $F \setminus u$ in one step, then $F \rightarrow_{L_o(t)} F, t$ implies $\ulcorner t[u \leftarrow \epsilon(u)] \urcorner \in forge(\ulcorner F[u \leftarrow \epsilon(u)] \urcorner)$ and $\epsilon(u) \in forge(F)$ for every message t.*

4 Main Theorem and the NP Decision Algorithm

We now state the main theorem of this paper. In Section 6, this theorem will allow us to show that INSECURE is in NP in presence of an intruder that uses XOR rules and prefix rules, respectively.

Theorem 1 *Let L_o be a set of oracle rules. If $E \rightarrow_{L_o} E, t$ can be checked in polynomial time in $|E, t|$ for every finite set E of messages and message t, then* INSECURE *is in NP.*

The NP decision procedure is given in Figure 1. We use the following notation and terminology. If Σ is a finite alphabet, then Σ^* shall denote the set of finite words over Σ. Let $\mathcal{S}$ denote a set of sets of intruder rules and $w \in \mathcal{S}^*$ with $w = \mathcal{L}_1 \cdots \mathcal{L}_n$, $\mathcal{L}_i \in \mathcal{S}$. Then, $\rightarrow_w$ is the composition $\rightarrow_{\mathcal{L}_1} \circ \cdots \circ \rightarrow_{\mathcal{L}_n}$; if w is the empty word, $\rightarrow_w$ is the identity relation. For some binary relation $\rightarrow$ on finite sets of messages, we say that $\rightarrow$ *generates* t from E if there exists an E' such that $E \rightarrow E'$ and $t \in E'$.

In (1) and (2) of the procedure, an attack (π, σ) is guessed of size linearly bounded in n. Then, it is checked whether this is in fact an attack. To this purpose, according to Definition 2, one has to check that $\ulcorner R_i\sigma \urcorner \in forge(\ulcorner S_0, S_1\sigma, ..., S_{i-1}\sigma \urcorner)$ for every $i \in \{1, \ldots, k\}$ and $Secret \in forge(\ulcorner S_0, S_1\sigma, ..., S_k\sigma \urcorner)$. This is done by guessing words l_i of length $\leq 5n$ over the alphabet $\{L_o(t), L_d(t), L_c(t) \mid t \text{ normalized and } |t| \leq 5n\}$. Note that every element $L_o(t)$, $L_d(t)$, and $L_c(t)$ in this alphabet can be represented in size linearly bounded in the size of P. Given the l_i's, one has to check that according to l_i the desired messages can actually be generated. This can be done in polynomial time, since, by assumption, $E \rightarrow_{L_o} E, t$ can be tested in polynomial time in $|E, t|$ and for $E \rightarrow_{L_d} E, t$ and $E \rightarrow_{L_c} E, t$ this is obvious. Thus, the procedure depicted in Figure 1 is in fact an NP procedure, and clearly, it is sound.

Now, to show that this procedure is also complete it obviously suffices to prove the following:

1. For every normalized finite set E of messages and normalized message t with $t \in forge(E)$, there exists a derivation D from E with goal t of length bounded by $|E, t|$ such that for every rule $L \in D$ if L is a t'-rule than $|t'| \leq |E, t|$.
2. If there exists an attack on P, then there exists an attack (π, σ) such that $\left|\ulcorner R_i\sigma, S_0\sigma, \ldots, S_{i-1}\sigma \urcorner\right| \leq 5 \cdot |P|$ and $\left|\ulcorner Secret, S_0\sigma, \ldots, S_k\sigma \urcorner\right| \leq 5 \cdot |P|$ for every $i \in \{1, \ldots, k\}$ (see Corollary 1).

The proof of the first statement is obvious: If $t \in forge(E)$, then Definition 4 implies that there exists a well formed derivation $D = E \rightarrow_{L_1} E, t_1 \rightarrow \ldots \rightarrow_{L_r} E, t_1, .., t_r$, with $t_r = t$. In particular, $t_i \in Sub(E, t)$ for every $i \in \{1, \ldots, k\}$. By definition of derivations, all t_i are different. It follows $r \leq |t, E|$.

A proof sketch of the second statement is given in the following section.

Input: protocol $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ with $n = |P|$, $V = Var(P)$.

1. Guess an execution order π for P. Let k be the size of π. Let $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$ for $i \in \{1, \ldots, k\}$
2. Guess a normalized ground substitution σ such that $|\sigma(x)| \leq 4n$ for all $x \in V$.
3. For all $i \in \{1, \ldots, k{+}1\}$ guess a word $l_i \in \{L_o(t), L_d(t), L_c(t) \mid t \text{ normalized and } |t| \leq 5n\}^*$ of length $\leq 5n$.
4. For all $i \in \{1, \ldots, k\}$, check whether $\rightarrow_{l_i}$ generates $\ulcorner R_i\sigma \urcorner$ from $\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner$.
5. Check that $\rightarrow_{l_{k+1}}$ generates $Secret$ from $\ulcorner \{S_j\sigma \mid j < k+1\} \cup \{S_0\} \urcorner$.
6. If each check is successful, then answer "yes", and otherwise, "no".

Figure 1: NP Decision Procedure for Insecurity

5 Linear Bounds on Attacks

To show Point 2. at the end of Section 4, we consider an attack of minimal size, a so-called normal attack, and show that this attack has the desired property.

In what follows, we assume that L_o is a set of oracle rules. If $t \in forge(E)$, we denote by $Deriv_t(E)$ a well formed derivation from E with goal t (chosen arbitrarily among the possible ones). Note that there always exists such a derivation since the definition of oracle rules ensures that a well formed derivation exists iff a derivation exists.

To define normal attacks, we need an ordering on multisets: We compare finite multisets on $\mathbb{N}$ by extending the usual ordering on $\mathbb{N}$ as follows: For finite multisets M and N on $\mathbb{N}$ define $M \gg N$ iff i) $M \neq N$ and *ii*) for every $x \in \mathbb{N}$, whenever $N(x) > M(x)$, then $M(y) > N(y)$ for some $y > x$. For instance, $\{3, 1, 1, 1\} \gg \{2, 2, 2, 1\}$.

Definition 5 *Let $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol. An attack (π, σ) is* normal *if the multiset of nonnegative integers $|\sigma| := \{|\sigma(x)| \mid x \in Var(P)\}$ is minimal w.r.t. $\gg$.*

Clearly, if there is an attack, there is a normal attack since $\gg$ is a well-founded ordering on finite multisets of nonnegative integers. Note also that a normal attack is not necessarily unique.

In Lemma 7 we prove, using Lemma 3 to 6, that normal attacks can always be constructed by linking subterms that are initially occurring in the problem specification. This will allow us to bound the size of attacks as desired (Theorem 2 and Corollary 1).

Let $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol such that (π, σ) is an attack on P. Let k be the size of π. We define $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$ for $i \in \{1, \ldots, k\}$, and $\mathcal{SP} = Sub(\{R_j | j \in \{1, \ldots, k\}\} \cup \{S_j | j \in \{0, \ldots, k\}\})$. We recall that $Atoms \subseteq \mathcal{SP}$. Let $Var = Var(\mathcal{SP})$, the set of variables occurring in the protocol.

Definition 6 *Let t and t' be two terms and θ a ground substitution. Then, t is a* θ-match *of t', denoted $t \sqsubseteq_\theta t'$, if t and t' are standard, t is not a variable, and $\ulcorner t\theta \urcorner = t'$.*

The proof of the following lemma can be found in Appendix 8.1.

Lemma 3 *If (π, σ) is a normal attack, then for all $i \in \{1, \ldots, k\}$, $x \in Var(R_i)$, and standard subterms s of $\sigma(x)$, there exists $j \leq i$ such that $s \in Sub(\ulcorner R_j\sigma \urcorner)$ or there exists $t \in \mathcal{SP}$ with $t \sqsubseteq_\sigma s$.*

The proof of the following lemma is trivial.

Lemma 4 *For every normalized finite set E of messages, message t, and t-rule L, if $E \rightarrow_L E, t$ then all proper subterms of t are subterms of E.*

Proof. For $L \in L_{od}$ use the definition of decomposition guess rules, for $L \in L_{oc}$ use Definition 4, (3), and for $L \in L_d \cup L_c$ the statement is obvious. □

The subsequent lemma will allow us to replace certain subterms occurring in a substitution of an attack by smaller terms (see Appendix 8.1 for the proof).

Lemma 5 *Let E and F be two sets of normalized messages such that $0 \in E \cup F$. Let $t \in forge(E, F)$ and $s \in forge(E)$ non atomic such that $s \notin Sub(E)$ and there exists a derivation D_s from E with goal s ending with an application of a rule in $\mathcal{L}_c$. Finally, let δ be the replacement $[s \leftarrow \epsilon(s)]$, where $\epsilon(s)$ is defined as in Definition 4. Then, $\ulcorner t\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.*

The next lemma will be used to remove one application of the normalization function.

Lemma 6 *Let σ be a normalized ground substitution, E a set of normalized terms, s a normalized standard non atomic term, and δ the replacement $[s \leftarrow \epsilon(s)]$. Let $\sigma' = \sigma\delta$. If there is no standard subterm t of E such that $t \sqsubseteq_\sigma s$, then $\ulcorner E\sigma' \urcorner = \ulcorner\ulcorner E\sigma \urcorner\delta \urcorner$.*

The main lemma, which shows that a substitution of a normal attack can be build up from subterms of terms occurring in P, is proved next.

Lemma 7 *Given a normal attack (π, σ), for all variables x and for all factors v_x of $\sigma(x)$, there exists $t \in \mathcal{SP}$ such that $t \sqsubseteq_\sigma v_x$.*

Proof. Assume that (*): For every t, $t \sqsubseteq_\sigma v_x$ implies $t \notin \mathcal{SP}$. We will lead this to a contradiction. Since $Atoms \subseteq \mathcal{SP}$, we have $v_x \notin Atoms$, and since v_x is a factor of $\sigma(x)$, v_x is standard. By Lemma 3 and (*), there exists j such that $v_x \in Sub(\ulcorner R_j\sigma \urcorner)$. Let N_x be minimal among the possible j. If $v_x \in Sub(\ulcorner S_i\sigma \urcorner)$ for some i, (*) implies that there exists $y \in Var(S_i)$ with $v_x \in Sub(\sigma(y))$. Then, by Definition 1, (2) there exists $R_{i'}, i' \leq i$ such that $y \in Var(R_{i'})$. Thus, Lemma 3 and (*) imply that there exists $j \leq i$ with $v_x \in Sub(\ulcorner R_j\sigma \urcorner)$. Note also that $v_x \notin Sub(S_0)$ since otherwise $v_x \in \mathcal{SP}$. Now, the minimality of N_x yields $i \geq N_x$. Summarizing, we have: v_x is not a subterm of $E_0 = \ulcorner S_0\sigma, \ldots, S_{N_x-1}\sigma \urcorner$, and v_x is a subterm of $\ulcorner R_{N_x}\sigma \urcorner$. Let us show now the following claim:

Claim. $v_x \in forge(E_0)$.
Proof of the claim. Let $Deriv_{\ulcorner R_{N_x}\sigma \urcorner}(E_0)$ be $E_0 \rightarrow_{L_1} E_1 ... \rightarrow_{L_n} E_n$. There exists a smallest $i \neq 0$ such that $v_x \in Sub(E_i)$ since v_x is subterm of $\ulcorner R_{N_x}\sigma \urcorner \in E_n$. If v_x is a proper subterm of some $s \in E_i$, Lemma 4 implies $v_x \in Sub(E_{i-1})$, a contradiction to the minimality of i. We conclude that $v_x \in E_i$, and therefore, $v_x \in forge(E_0)$. This finishes the proof of the claim.

Let us define the replacement $\delta = [v_x \leftarrow \epsilon(v_x)]$ where $\varepsilon(v_x)$ is defined as in Definition 4. Since (π, σ) is an attack, for all j, we have:

$$\ulcorner R_j\sigma \urcorner \in forge(\ulcorner S_0\sigma, \ldots, S_{j-1}\sigma \urcorner)$$

We distinguish two cases:

- Assume $j < N_x$. By minimality of N_x, v_x is neither a subterm of $\ulcorner R_j\sigma \urcorner$ nor a subterm of $\ulcorner S_0\sigma, \ldots, S_{j-1}\sigma \urcorner$. Hence, with $\ulcorner R_j\sigma \urcorner \in forge(\ulcorner S_0\sigma, \ldots, S_{j-1}\sigma \urcorner)$ it follows $\ulcorner\ulcorner R_j\sigma \urcorner\delta \urcorner \in forge(\ulcorner\ulcorner S_0\sigma \urcorner\delta, \ldots, \ulcorner S_{j-1}\sigma \urcorner\delta \urcorner)$.
- Assume $j \geq N_x$. The last rule in $Deriv_{v_x}(\ulcorner S_0\sigma, \ldots, S_{N_x-1}\sigma \urcorner)$ is in $\mathcal{L}_c$ since otherwise v_x is a subterm of $\ulcorner S_0\sigma, \ldots, S_{N_x-1}\sigma \urcorner$. Now, with $E = E_0$ and $F = \ulcorner S_{N_x}\sigma, \ldots, S_{j-1}\sigma \urcorner$, Lemma 5 implies $\ulcorner\ulcorner R_j\sigma \urcorner\delta \urcorner \in forge(\ulcorner\ulcorner S_0\sigma \urcorner\delta, \ldots \ulcorner S_{j-1}\sigma \urcorner\delta \urcorner)$.

Thus, $\ulcorner R_j\sigma\urcorner\delta\urcorner \in forge(\ulcorner S_0\sigma\urcorner\delta, \ldots \ulcorner S_{j-1}\sigma\urcorner\delta\urcorner)$ in both cases. Now, with $E = \{S_0, \ldots, S_{j-1}\}$ and $E = \{R_j\}$, respectively, (*) and Lemma 6 imply for all j:

$$\ulcorner R_j\sigma'\urcorner \in forge(\ulcorner S_0\sigma', \ldots, S_{j-1}\sigma'\urcorner)$$

where $\sigma' = \sigma\delta$. Hence, (π, σ') is an attack. But since σ' is obtained from σ by replacing v_x by a strictly smaller message, namely $\epsilon(v_x)$, we obtain $|\sigma'| \ll |\sigma|$, a contradiction to the assumption that (π, σ) is a normal attack. □

We can now use this lemma to bound the size of every $\sigma(x)$:

Theorem 2 *For every protocol P, if (π, σ) is a normal attack on P, then $|\{\sigma(x) \mid x \in Var\}| \leq 4 \cdot |P|$, where $|P|$ is the size of P as defined in Section 3.2.*

Proof. Let $F = \{s \mid \exists x \in Var,\ s \in Factor(\sigma(x))\}$ For every s in the set F we introduce a new variable x_s and we define a substitution σ' such that $\sigma'(x_s) = s$ (and other variables are mapped to themselves). Let $Var' = \{x_s\}_{s\in F}$. The cardinality of Var' can be bounded as follows:

Claim. $|Var'| \leq |P|$
Proof of the claim. We define a function $f : Var' \to \mathcal{SP}$ as follows. Due to Lemma 7, for every $y \in Var'$, there exists $t_y \in \mathcal{SP}$ such that $t_y \notin Var$ and $\ulcorner t_y\sigma\urcorner = \sigma'(y)$. We define $f(y) := t_y$. The function f is injective since $t_s = t_{s'}$ implies $\ulcorner t_s\sigma\urcorner = \ulcorner t_{s'}\sigma\urcorner$. Thus, $|Var'| \leq |\mathcal{SP}| = |P|$, which concludes the proof of the claim.

Let $S = F \cup \{\sigma(x) \mid x \in Var\}$. For all $x \in Var$ let $\sigma''(x) = \ulcorner \text{XOR}(x_{s_1}, \ldots, x_{s_m})\urcorner$ with $\{s_1, \ldots, s_m\} = Factor(\sigma(x))$. Note that, since the s's are normalized standard messages, $\sigma(x) = \sigma'(\sigma''(x))$. Let δ be the composition of all replacements $[s \leftarrow x_s]$, $s \in F$ (replacing larger terms before), and let $t\varphi := (t\sigma'')\delta$. It is not hard to see that that $\ulcorner t\varphi\urcorner\sigma' = \ulcorner t\sigma\urcorner$.

We are now going to bound $|S|$. Given a set of normalized messages Z, let

$$V_Z = \{x \in Var \mid \sigma(x) \text{ non standard and } \sigma(x) \notin Z\},$$
$$P_Z = \{\ulcorner t\varphi\urcorner \mid t \in \mathcal{SP} \text{ and } \ulcorner t\sigma\urcorner \notin Z\}.$$

We note that $Z \subseteq Z'$ implies $V_{Z'} \subseteq V_Z$ and $P_{Z'} \subseteq P_Z$, and that $V_S = \emptyset$.

Claim. $|S \cup P_S| \leq |V_\emptyset \cup P_\emptyset|$.
Proof of the claim. We construct a sequence of sets $S = Z_1 \supset Z_2 \supset \cdots \supset Z_n = \emptyset$ with $Z_{i+1} = Z_i \setminus v_i$ where $v_i \in Z_i$ is a maximal message in Z_i (w.r.t. the subterm ordering). Note that $n-1$ is the cardinality of S and for every $t \in Z_{i+1}$, $v_i \notin Factor(t)$. For every $i \in \{1, \ldots, n\}$ we prove

$$|Z_i \cup V_{Z_i} \cup P_{Z_i}| \leq |Z_{i+1} \cup V_{Z_{i+1}} \cup P_{Z_{i+1}}|$$

which concludes the proof of the claim. At step i, either of two cases may arise when removing $v = v_i \in Z_i$ from Z_i:

- There exists $x \in Var$ with $v = \sigma(x)$ non standard. Then,

$$\begin{aligned} &|Z_i \cup V_{Z_i} \cup P_{Z_i}| \\ &\quad \leq |Z_i \setminus v \cup \{x\} \cup Factor(\sigma(x)) \cup V_{Z_i} \cup P_{Z_i}| \\ &\quad \leq \left|Z_{i+1} \cup V_{Z_{i+1}} \cup P_{Z_{i+1}}\right| \end{aligned}$$

  since $x \notin Z_i \cup V_{Z_i}$, $x \in V_{Z_{i+1}}$, and $Factor(\sigma(x)) \subseteq Z_i \setminus v = Z_{i+1}$.

- $v \in F$ and there exists $t \in \mathcal{SP}$ such that $t \sqsubseteq_\sigma v$. Let $t' = \ulcorner t\varphi\urcorner$. We have $t'\sigma' = \ulcorner t\sigma\urcorner = v$. Then,

$$\begin{aligned} |Z_i \cup V_{Z_i} \cup P_{Z_i}| \ &\leq \left|Z_{i+1} \cup V_{Z_{i+1}} \cup \{t'\} \cup P_{Z_i}\right| \\ &\leq \left|Z_{i+1} \cup V_{Z_{i+1}} \cup P_{Z_{i+1}}\right| \end{aligned}$$

  since $\sigma'(y) \in Z_i \setminus v = Z_{i+1}$ for every $y \in Var(t')$ and $P_{Z_{i+1}} = P_{Z_i} \cup \{t'\}$.

This proves the claim. Using the claim and

$$\begin{aligned} |P_\emptyset| &= \left|\ulcorner \mathcal{S}P\varphi \urcorner\right| \leq |\mathcal{S}P\varphi| \leq |\mathcal{S}P| + |Var\varphi|\,, \text{ and} \\ |Var\varphi| &= |Var\sigma''| \leq |Var| + |Var'|\,, \end{aligned}$$

we obtain

$$\begin{aligned} |\{\sigma(x)\,|\,x \in Var\}| &\leq |S| \leq |S \cup P_S| \leq |V_\emptyset \cup P_\emptyset| \\ &\leq 2 \cdot |Var| + |\mathcal{S}P| + |Var'| \leq 4 \cdot |P| \end{aligned}$$

□

From this, statement 2. at the end of Section 4 easily follows:

Corollary 1 *For every protocol P and normal attack (π, σ) on P, and for every $i \in \{1, \ldots, k\}$ with S_j and R_j as defined above :*

$$|R_i\sigma, S_0\sigma, \ldots, S_{i-1}\sigma| \leq 5 \cdot |P| \text{ and } |Secret, S_0\sigma, \ldots, S_k\sigma| \leq 5 \cdot |P|$$

6 Extending the Dolev-Yao Intruder by Different Oracle Rules

We extend the ability of the standard Dolev-Yao intruder beyond the perfect encryption hypothesis by considering two specific sets of oracle rules. The first set are the XOR rules which allow the intruder to make use of the XOR operator. We then consider, what we call, prefix rules which allow the intruder to exploit certain properties of encryption based on block ciphers.

6.1 XOR Rules

The XOR rules allow the intruder to sum several messages with the XOR operator. The result of this sum is being normalized.

Definition 7 *We define $L_o = L_{oc} \cup L_{od}$ to be the set of* XOR rules *where*

- *L_{oc} is the set of rules of the form $\{t_1, \ldots, t_n\} \to \ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ with $\{t_1, \ldots, t_n\}$ a non-empty finite multiset of normalized messages such that $\ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ is non-standard, and*
- *L_{od} is the set of rules of the form $\{t_1, \ldots, t_n\} \to \ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ with $\{t_1, \ldots, t_n\}$ a non-empty finite multiset of normalized messages such that $\ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ is standard.*

We call the intruder using the rules $L_o \cup L_c \cup L_d$ the XOR intruder.

Note that the rules in L_{od} are in fact decomposition guess rules since if $\ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ is standard, it is a subterm of $t_1, \ldots, t_n$.

We also note that the intruder is not more powerful if we allow him to derive non-normalized messages. More precisely, assume that L_e is the set of rules of the form $\{t_1, \ldots, t_n\} \to s$ with $s =_{XOR} \text{XOR}(t_1, \ldots, t_n)$ (not necessarily normalized). Let $forge_e(E)$ denote the set of messages the intruder can derive from E with the rules L_e, L_d, and L_c. Then, it easily follows by induction on the length of derivations:

Proposition 1 *For every message term t and set of messages E (both not necessarily normalized), $t \in forge'(E)$ implies $\ulcorner t \urcorner \in forge(\ulcorner E \urcorner)$.*

Therefore, we can restrict the intruder to work only on normalized messages and to produce only normalized messages.

Before showing that the XOR rules are oracle rules, we illustrate that the XOR intruder can perform the attack informally described in Section 2.

Formally, the protocol underlying the attack is described as follows: The initial intruder knowledge is $S_0 = \{0, I, ki, ki', ka, kb\}$, with $Atoms = \{na, a, I, b, ka, kb, ki, ki^{-1}, 0, secret\}$ the set of atoms (in Section 2 the secret

$$\begin{array}{lrcl} (a,1): & 0 & \Rightarrow & \{\langle na, a\rangle\}^p_{ki} \\ (a,2): & \{\langle x_{secret}, \text{XOR}(na, I)\rangle\}^p_{ka} & \Rightarrow & \{x_{secret}\}^p_{ki} \\ (b,1): & \{\langle x_{na}, a\rangle\}^p_{kb} & \Rightarrow & \{\langle secret, \text{XOR}(x_{na}, b)\rangle\}^p_{ka} \end{array}$$

- $\{\langle \text{XOR}(na, b, I), a\rangle\}^p_{kb} \in forge(0, I, ki, ki', ka, kb, \{\langle na, a\rangle\}^p_{ki})$;
- $\{\langle secret, \text{XOR}(na, I)\rangle\}^p_{ka} \in forge(0, I, ki, ki', ka, kb, \{\langle na, a\rangle\}^p_{ki}, \ulcorner\{\langle secret, \text{XOR}(\text{XOR}(na, b, I), b)\rangle\}^p_{ka}\urcorner)$;
- $secret \in forge(0, I, ki, ki', ka, kb, \{\langle na, a\rangle\}^p_{ki}, \ulcorner\{\langle secret, \text{XOR}(\text{XOR}(na, b, I), b)\rangle\}^p_{ka}\urcorner, \{secret\}^p_{ki})$.

Figure 2: An attack on the modified NSL protocol

was N_B). The protocol rules are depicted at the top of Figure 2. We have $\mathcal{I} = \{(a,1),(a,2),(b,1)\}$ and $<_{\mathcal{I}} := \{((a,1),(a,2))\}$.

When using a perfect encryption model, there is no attack on this instance of the protocol since the intruder is not able to forge $\{secret, \text{XOR}(na1, I)\}^p_{ka}$ without the oracle rules.

On the other hand, when using these rules, (π, σ) with the execution order $\pi = \{(a,1) \mapsto 0, (b,1) \mapsto 1, (a,2) \mapsto 2\}$ and the substitution σ with $\sigma(x_{na}) = \text{XOR}(na, b, I)$ and $\sigma(x_{secret}) = secret$ is an attack on this protocol. In fact, it is easy to check the steps depicted at the bottom of Figure 2.

We can show (see details in Appendix 8.2):

Proposition 2 *The set L_o of XOR rules is a set of oracle rules.*

Also, we can show that XOR rules can be applied in polynomial time.

Proposition 3 *Let L_o be the set of XOR rules. Then, the problem whether $E \rightarrow_{L_o(t)} E, t$, for a given finite normalized set E of messages and a normalized message t, can be decided in polynomial time with respect to $|E, t|$.*

As an immediate consequence of Theorem 1 we obtain that INSECURE with XOR rules is in NP. NP-hardness can be obtained as in [21]. Altogether this yields:

Theorem 3 INSECURE *w.r.t. the XOR intruder is an NP-complete problem.*

6.2 Prefix Rules

As another instance of oracles rules, we consider what we call prefix rules. These rules allow the intruder to exploit certain properties of block encryption algorithms, based for example on cipher block chaining (CBC). Using Theorem 1, again we can show that INSECURE is an NP-complete problem. In Appendix 8.3 an example that illustrates the additional power of the intruder is provided.

Throughout this section, we assume that terms do not contain the XOR operator and that the normalization function $\ulcorner\cdot\urcorner$ is the identity function. It is easy to verify that Theorem 1 also holds in this simplified setting.

Definition 8 *We define $L_o = L_{oc} \cup L_{od}$ to be the set of* prefix rules *where $L_{od} = \emptyset$ and L_{oc} consists of intruder rules of the form*

$$\{\langle\langle \ldots \langle\langle M, M_1\rangle, M_2\rangle, \ldots\rangle, M_n\rangle\}^s_K \rightarrow_{L_{oc}} \{M\}^s_K$$

for any normalized messages $K, M, M_1, \ldots, M_n$, ($n \geq 1$). We call the intruder using the rule $L_o \cup L_c \cup L_d$ prefix intruder.

We can prove that these *prefix* rules are oracle rules that can be checked in polynomial time and then conclude that INSECURE for an intruder equipped with prefix rules is NP-complete by Theorem 1.

Proposition 4 *The set L_o of prefix rules is a set of oracle rules.*

Obviously, $E \rightarrow_{\mathcal{L}_c} E, t$ can be decided in polynomial time in $|E, t|$. Also, analogously to the proof in [21] one can show that INSECURE is NP-hard. Now, by Theorem 1, it follows:

Theorem 4 INSECURE *w.r.t. the prefix intruder is an NP-complete problem.*

7 Conclusion

We have shown that when extending the standard Dolev Yao intruder by rules for XORing messages the protocol insecurity problem remains NP-complete for finite sessions. To our knowledge there is no published result on deciding insecurity without the perfect encryption hypothesis. Here we have only considered insecurity as failure of secrecy. However, we believe that our result holds also for other properties (e.g. authentification) that can be reduced to reachability problems in our model. Future work includes applying our approach to different intruder rules and to algebraic laws such as the ones relying on RSA and Diffie-Hellman encryption techniques.

References

[1] R. Amadio and W. Charatonik. On name generation and set-based analysis in the Dolev-Yao model. In *Proceedings of CONCUR 02*, number 2421 in Lecture Notes in Computer Science, pages 499–512. Springer-Verlag, 2002.

[2] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proceedings of CONCUR'00*, volume 1877 of *Lecture Notes in Computer Science*, 2000.

[3] D. Basin. Lazy infinite-state analysis of security protocols. In R. Baumgart, editor, *Secure Networking — CQRE (Secure)'99*, LNCS 1740, pages 30–42. Springer-Verlag, 1999.

[4] R. Book and F. Otto. *String-rewriting systems*. Springer, New York, 1993.

[5] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th International Conference on Automata, Language and Programming: ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.

[6] Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In E. Brinksma and K. Guldstrand Larsen, editors, *14th International Conference on Computer Aided Verification, CAV'2002*, volume 2404 of *Lecture Notes in Computer Science*, pages 324–337, Copenhagen (Denmark), July 2002. Springer.

[7] J. Clark and J. Jacob. A survey of authentication protocol literature. `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`, 1997.

[8] H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping pong protocols. In *Proceedings of the 28th International Conference on Automata, Language and Programming: ICALP'01*, LNCS 2076, Berlin, 2001. Springer-Verlag.

[9] D. Dolev and A. Yao. On the security of public key protocols. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 350–357, 1981.

[10] N. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of Bounded Security Protocols. In *Proceedings of the FLOC'99 Workshop on Formal Methods and Security Protocols*, 1999.

[11] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.

[12] R. Küsters. On the decidability of cryptographic protocols with open-ended data structures. In L. Brim, P. Jancar, M. Kretinsky, and A. Kucera, editors, *13th International Conference on Concurrency Theory (CONCUR 2002)*, volume 2421, pages 515–530. Springer-Verlag, 2002.

[13] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1996.

[14] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.

[15] C. Meadows and P. Narendran. A unification algorithm for the group Diffie-Hellman protocol. In *Workshop on Issues in the Theory of Security (in conjunction with POPL'02), Portland, Oregon, USA, January 14-15*, 2002.

[16] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[17] J. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, 1998.

[18] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Technical Report CSL-78-4, Xerox Palo Alto Research Center, Palo Alto, CA, USA, 1978. Reprinted June 1982.

[19] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, 1997.

[20] O. Pereira and J.-J. Quisquater. On the perfect encryption assumption. In *Workshop on Issues in the Theory of Security (in conjunction with ICALP'00), University of Geneva, Switzerland 7,8 July 2000*, pages 42–45, 2000.

[21] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001. long version to appear in Theoretical Computer Science.

[22] P. Ryan and S. Schneider. An attack on a recursive authentication protocol. *Information Processing Letters 65*, 1998.

[23] D. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.

8 Appendix

8.1 Bounds on Attacks

We give the remaining proofs for the lemmas needed in order to derive polynomial bounds on normal attacks

Lemma 3 *Given a normal attack (π, σ) on P, for all $i \in \{1, \ldots, k\}$, $x \in Var(R_i)$, and s standard subterm of $\sigma(x)$, either there exists $j \leq i$ such that $s \in Sub(\ulcorner R_j\sigma \urcorner)$ or there exists $t \in \mathcal{SP}$ with $t \sqsubseteq_\sigma s$.*

Proof. Assume that there exists $i \in \{1, \ldots, k\}$, $x \in Var(R_i)$, and s standard subterm of $\sigma(x)$ such that for all $j \leq i$: $s \notin Sub(\ulcorner R_j\sigma \urcorner)$. Define

$$j = \min\{i' \mid y \in Var(R_{i'}) \text{ and } s \text{ subterm of } \sigma(y)\}$$

and let y be a variable of R_j such that s is a subterm of $\sigma(y)$. Note that $j \leq i$, and thus, $s \notin Sub(\ulcorner R_j\sigma \urcorner)$. Let $S_{y,s}$ be the set of subterms t of R_j such that $y \in Var(t)$ and s is a subterm of $\ulcorner t\sigma \urcorner$. This subset contains y, and thus, is not empty. Let $t \in S_{y,s}$ be maximal in $S_{y,s}$ w.r.t. the subterm ordering. We know that $t \neq R_j$. Let $r \in Sub(R_j)$ with $t \in Sub(r)$ and there exists no $r' \in Sub(r)$ with $t \in Sub(r')$. Then, since $s \notin Sub(\ulcorner r\sigma \urcorner)$, r must be of the form $\text{XOR}(t, t_1, \ldots, t_n)$ with $t, t_1, \ldots, t_n$ standard (since R_j is normalized) and $n \geq 1$ such that there exists $i \in \{1, \ldots, n\}$, say $i = 1$, with $\ulcorner t\sigma \urcorner = \ulcorner t_1\sigma \urcorner$ (s has been eliminated by normalization). In particular, $s \in Sub(\ulcorner t_1\sigma \urcorner)$.

Let M_{s,t_1} be the set of subterms t' of t_1 such that $s \in Sub(\ulcorner t'\sigma \urcorner)$, and let t_s be minimal in M_{s,t_1} w.r.t. the subterm ordering. By Definition 1, (3), and since y first appears in R_j, we have that for all $z \in Var(t_1)$, there exists $j_z < j$ with $z \in Var(R_{j_z})$. Hence, by minimality of j, s is not a subterm of $\{\sigma(z) | z \in Var(t_1))\}$, and thus, $t_s \notin Var$. Thus, by minimality of t_s, $\ulcorner t_s\sigma \urcorner = s$. Moreover, t_s is standard by minimality (otherwise, since s is standard, there would be a factor t'_s of t_s such that $s \in Sub(\ulcorner t'_s\sigma \urcorner)$). Together, this implies $t_s \sqsubseteq_\sigma s$. □

Lemma 4 *For every normalized finite set E of messages, message t, and t-rule L, if $E \rightarrow_L E, t$ then all proper subterms of t are subterms of E.*

Proof. For $L \in L_{od}$ use the definition of decomposition guess rules, for $L \in L_{oc}$ use Definition 4, (3), and for $L \in L_d \cup L_c$ this is obvious. □

This Lemma 4 will be used to prove Lemma 5. Roughly speaking, the following Lemma 8, to be also used in the proof of Lemma 5, states that if a term γ can be forged from a set of messages E by composing with L_c, say composing two messages γ_1 and γ_2 both derived from E, then it is always possible to avoid decomposing γ with L_d in a derivation from E with goal t for some t since such a decomposition would generate a message γ_1 or γ_2 that can be derived from E in another way.

First, we need some notation: If $D_1 = E_1 \rightarrow \ldots \rightarrow F_1$ and $D_2 = E_2 \rightarrow \ldots \rightarrow F_2$ are two derivations such that $E_2 \subseteq F_1$, then $D = D_1.D_2$ is defined as the *concatenation* of the steps of D_1 and the ones in D_2. In addition, to obtain a derivation, we delete in D the steps from D_2 that generate terms already present in F_1.

Lemma 8 *Let $t \in forge(E)$ and $\gamma \in forge(E)$ be given with a derivation D_γ from E ending with an application of a rule in $\mathcal{L}_c$. Then, there is a derivation D' from E with goal t satisfying $L_d(\gamma) \notin D'$.*

Proof. By definition of a derivation, $L_d(\gamma) \notin D_\gamma$. Let D be D_γ without its last rule, i.e., D_γ is D followed by some $L \in \mathcal{L}_c$. Define $D'' = D.Deriv_t(E) = D.D'''$ — D''' is obtained from $Derive_t(E)$ by removing redundant steps. Note that D'' is a derivation with goal t. We distinguish two cases:

- Assume $L = L_c(\gamma)$. Then $L_d(\gamma) \notin D''$ since the (two) direct subterms of γ are created in D, and thus, $L_d(\gamma) \notin D''$. In other words, $D' = D''$ is the derivation we are looking for.
- Assume $L = L_{oc}(\gamma)$. Then, if $L_d(\gamma) \notin D''$ setting $D' = D''$ we are done. Otherwise, let F_1 be the final set of messages of D. Now, Definition 4, (2) implies that every step in D''' of the form $F_1, F_2, \gamma \rightarrow_{L_d(\gamma)} F_1, F_2, \gamma, \beta$ can be replaced by a derivation from F_1, F_2 with goal β that does not contain rules from $L_d(\gamma)$. Replacing steps in this way and then removing redundant steps yields the derivation D' we are looking for.

□

Lemma 5 *Let E and F be two sets of normalized messages such that $0 \in E \cup F$. Let $t \in forge(E, F)$ and $s \in forge(E)$ non atomic such that $s \notin Sub(E)$ and there exists a derivation D_s from E with goal s ending with an application of a rule in $\mathcal{L}_c$. Finally, let δ be the replacement $[s \leftarrow \epsilon(s)]$, where $\epsilon(s)$ is defined as in Definition 4. Then, $\ulcorner t\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.*

Proof. Let $D_s = E \rightarrow_{L_1} E, t_1 \rightarrow_{L_2} \ldots \rightarrow_{L_p} E, t_1, .., t_p \rightarrow_{\mathcal{L}_c(s)} E, t_1, .., t_p, s$. By induction on i and using Lemma 4, it follows that all proper subterms of t_i are subterms of $E, t_1, .., t_{i-1}$. Using $s \notin Sub(E)$ and $s \neq t_i$, this implies $s \notin Sub(t_i)$, and thus, $\ulcorner t_i\delta \urcorner = t_i$ (note that t_i is normalized) and $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta \urcorner)$, for every $i \in \{1, \ldots, p\}$. Thanks to Lemma 8, there exists a derivation $D = E, F, t_1, .., t_p \rightarrow_{L_{p+1}} E, F, t_1, .., t_{p+1} \rightarrow \ldots \rightarrow_{L_n} E, F, t_1, .., t_n$ with $t_n = t$ and $L_i \notin L_d(s)$, for every $i \in \{p+1, \ldots, n\}$. We know $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta \urcorner)$, for every $i \in \{1, \ldots, p\}$. We show by induction on i, $p \leq i \leq n$, that $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$. For $i = p$ this is by Definition 4, (4). Assume that $i > p$ and the property is true for all $j < i$. Then we have three cases:

- If $L_i = L_c(\langle a, b\rangle)$, then either $t_i = s$, and thus, using the definition of $\epsilon(s)$, $\ulcorner t_i\delta \urcorner = \epsilon(s) \in forge(\ulcorner E\delta, F\delta \urcorner)$, or $\ulcorner t_i\delta \urcorner = \langle \ulcorner a\delta \urcorner, \ulcorner b\delta \urcorner \rangle \in forge(\ulcorner E\delta, F\delta \urcorner)$ since $\{a, b\} \subseteq E \cup F \cup \{t_1, .., t_{i-1}\}$. Analogously for $\{a\}^s_b$ and $\{a\}^p_K$.
- If $L_i = L_{p1}(\langle t_i, a\rangle)$, then $s \neq \langle t_i, a\rangle$ since $L_i \notin L_d(s)$. Therefore, $\ulcorner t_i\delta \urcorner \in forge(\ulcorner \langle t_i, a\rangle\, \delta \urcorner) \subseteq forge(\ulcorner E\delta, F\delta \urcorner)$ since $\langle t_i, a\rangle \in E \cup F \cup \{t_1, .., t_{i-1}\}$. Analogously for L_{p2}, L_{sd}, and L_{ad}.
- If $L_i \in L_{oc} \cup L_{od}$, then thanks to Definition 4, (4), we have: $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta, t_1\delta, .., t_{i-1}\delta \urcorner)$ and thus: $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.

For $i = n$, this gives us $\ulcorner t\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$. □

Lemma 6 *Let σ be a normalized ground substitution, E a set of normalized terms, s a normalized standard non atomic term, and δ the replacement $[s \leftarrow \epsilon(s)]$. Let $\sigma' = \sigma\delta$. If there is no standard subterm t of E such that $t \sqsubseteq_\sigma s$, then $\ulcorner E\sigma' \urcorner = \ulcorner\ulcorner E\sigma \urcorner\delta \urcorner$.*

Proof. Since there is no standard subterm t' of E such that $t' \sqsubseteq_\sigma s$, we have $(E\sigma)\delta = E(\sigma\delta)$ and therefore $\ulcorner E\sigma' \urcorner = \ulcorner (E\sigma)\delta \urcorner$. Let us prove, by induction on the structure of terms, that for all $t \in Sub(E)$, we have $\ulcorner t\sigma' \urcorner = \ulcorner\ulcorner t\sigma \urcorner\delta \urcorner$. This will conclude the proof of the lemma.

- If $t \in Atoms$, then $t \neq s$ by assumption. Thus, $\ulcorner\ulcorner t\sigma \urcorner\delta \urcorner = t = \ulcorner t\sigma' \urcorner$.
- If $t \in Var$, then $\ulcorner t\sigma \urcorner = t\sigma$, and therefore, $\ulcorner t\sigma' \urcorner = \ulcorner (t\sigma)\delta \urcorner = \ulcorner\ulcorner t\sigma \urcorner\delta \urcorner$.
- If $t = \langle v, w\rangle$, we have $s \neq \langle \ulcorner v\sigma \urcorner, \ulcorner w\sigma \urcorner \rangle$ since otherwise $t \sqsubseteq_\sigma s$, and $\ulcorner t\sigma' \urcorner = \langle \ulcorner v\sigma' \urcorner, \ulcorner w\sigma' \urcorner \rangle$. By induction, this gives $\ulcorner t\sigma' \urcorner = \langle \ulcorner\ulcorner v\sigma \urcorner\delta \urcorner, \ulcorner\ulcorner w\sigma \urcorner\delta \urcorner \rangle$, and therefore, $\ulcorner t\sigma' \urcorner = \ulcorner\ulcorner \langle v\sigma, w\sigma \rangle \urcorner\delta \urcorner = \ulcorner\ulcorner t\sigma \urcorner\delta \urcorner$ since $s \neq \ulcorner t\sigma \urcorner$. The cases $t = \{u\}^s_v$ and $t = \{u\}^p_K$ are similar.
- If $t = \text{XOR}(\mathcal{T})$, where $\mathcal{T}$ is a multiset of standard terms, we have:

$$\begin{aligned}
\ulcorner t\sigma' \urcorner &= \ulcorner \text{XOR}(\{t'\sigma' \mid t' \in \mathcal{T}\}) \urcorner \\
&= \ulcorner \text{XOR}(\{\ulcorner t'\sigma' \urcorner \mid t' \in \mathcal{T}\}) \urcorner \\
&= \ulcorner \text{XOR}(\{\ulcorner\ulcorner t'\sigma \urcorner\delta \urcorner \mid t' \in \mathcal{T}\}) \urcorner \quad \text{(by induction)} \\
&= \ulcorner \text{XOR}(\{\ulcorner t'\sigma \urcorner\delta \mid t' \in \mathcal{T}\}) \urcorner \\
&= \ulcorner \text{XOR}(\{\ulcorner t'\sigma \urcorner \mid t' \in \mathcal{T}\})\delta \urcorner \\
&= \ulcorner\ulcorner t\sigma \urcorner\delta \urcorner
\end{aligned}$$

□

8.2 XOR rules

We now prove that the XOR rules form a set of oracle rules. We start to show Definition 4, (1). To do so, we first prove a sufficient condition for a derivation to be well formed.

Lemma 9 *Let $D = E_0 \rightarrow_{L_1} \dots E_{n-1} \rightarrow_{L_n} E_n$ be a derivation with goal g such that:*

1. *For every j with $E_{j-1} \rightarrow_{L_j} E_{j-1}, t$ the jth step in D and $L_j \in \mathcal{L}_d(t)$, there exists $t' \in E_{j-1}$ such that t is a subterm of t' and either $t' \in E_0$ or there exists i with $i < j$ and $L_i \in \mathcal{L}_d(t')$.*

2. *For every $i < n$ and t with $L_i \in \mathcal{L}_c(t)$, there exists j with $i < j$ such that L_j is a t'-rule and $t \in Sub(\{t'\} \cup E)$.*

Then, D is a well formed derivation with goal g.

Proof. From (1) it immediately follows by induction on $i \in \{1, \dots, n\}$ that $L_i \in \mathcal{L}_d(t)$ implies $t \in Sub(E_0)$ for every message t.

Using (2), we prove by induction on $n - i$ that for all $i \in \{1, \dots, n\}$, $L_i \in \mathcal{L}_c(t)$ implies $t \in Sub(E_0, g)$. If $n - i = 0$, then $t = g$ and therefore $t \in Sub(E_0, g)$. For the induction step, (2) implies that there exists $j > i$ such that L_j is a t'-rule and $t \in Sub(E_0, t')$. If $L_j \in \mathcal{L}_d(t')$, then $t' \in Sub(E_0)$ (see above). If $L_j \in \mathcal{L}_c(t')$, then by induction $t' \in Sub(E_0, g)$, and hence, $t \in Sub(E_0, g)$. □

Now, we can prove that XOR rules admit well formed derivations.

Proposition 5 *For every finite normalized set E of messages and normalized message g, $g \in forge(E)$ implies that there exists a well formed derivation from E with goal g.*

Proof. Let $E_0 = E$ and $D = E_0 \rightarrow_{L_1} \dots \rightarrow_{L_n} E_n$ be a derivation of goal g of minimal length. We prove that D satisfies (1) and (2) in Lemma 9. We first show:

Claim. If $F \rightarrow_{L_o(t)} F, t \rightarrow_{L_o(u)} F, t, u$, then $F \rightarrow_{L_o(u)} F, u$.
Proof of the claim. By definition of XOR rules, u is a normalized XOR sum of elements in F, t and t is a normalized XOR sum of elements in F. Thus, u is an XOR sum of elements in F. Thus, $F \rightarrow_{L_o(u)} F, u$. This concludes the proof of the claim.

By the claim w.l.o.g. we may assume that in D the terms used on the left hand-side of an XOR rules are not generated by XOR rules. Formally (*): For every i with $L_i \in L_o(t)$ and $L_i = F \rightarrow t$, there does not exist $j \in \{1, \dots, n\}$ such that $L_j \in L_o(t')$ for some $t' \in F$.
Now, we prove (1) and (2) of Lemma 9:

1. If $L_j \in L_d(s) \cap \mathcal{L}_d(t)$, then $L_i \notin L_{oc}(s)$, for all $i < j$, since rules in L_{oc} do not create standard terms, and $L_i \notin L_c(s)$, for all $i < j$, by the definition of derivation (since otherwise t would be in the left-hand side of L_i). Therefore, either $s \in E$ or there exists $i < j$ with $L_i \in \mathcal{L}_d(s)$.

   If $L_j \in L_{od}(t)$, then t is standard and, by (*) and the definition of L_{od}, there exists a non standard term t' in E_{j-1} with t subterm of t' and such that $L_{oc}(t') \notin D$. If $t' \in E$, we are done. Otherwise, there exists $i < j$ such that L_i is a t'-rule. Since t' is non standard, $L_j \notin L_c \cup L_{od}$, and by (*), $L_j \notin L_{oc}$. Thus, $L_j \in L_d$.

2. If $L_i \in L_c(t)$ and $i < n$, then t is standard, and by minimality of D, there exists $j > i$ such that t belongs to the left-hand side of L_j. By definition of a derivation, $L_j \notin L_d(t)$. If $L_j \in L_c(t')$, then $t \in Sub(t')$, and if $L_j \in L_o(t')$, then since t is standard, either t is a factor of t', and thus, $t \in Sub(t')$, or there exists $t'' \in E_{j-1}$ non standard with $t \in Sub(t'')$ (t is used to simplify t''). By (*), t'' was not generated by some rule in L_o. Since t'' is non standard it cannot be generated by some rule in L_c. Thus, either t'' was generated by L_d or $t'' \in E_0$. In the latter case, we are done. In the former case, 1. implies $t'' \in E_0$ as well.

   If $L_i \in L_{oc}(t)$ and $i < n$, then (*) implies that $t = g$ or there exists $j > i$ such that $L_i \in L_c(t')$ and $t \in Sub(t')$.

□

With this we can show:

Proposition 2 *The set L_o of XOR rules is a set of oracle rules.*

Proof. We check each condition in Definition 4:

1. The first point is a consequence of Proposition 5.
2. No term created with L_{oc} can be decomposed with L_d.
3. For $F \to s \in L_{oc}(s)$, every proper subterm of s is a subterm of F by the definition of L_{oc}.
4. For every non-atomic message u define $\epsilon(u) = 0$. Let u be a non-atomic message, F be a set of messages with $0 \in F$ and t be a message such that $F \setminus u \to_{\mathcal{L}_c(u)} F$ and $F \to_{L_o(t)} F, t$. Obviously, $\epsilon(u) \in forge(F)$. Let $\delta := [u \leftarrow 0]$. There are three cases:
   (a) Either $u = t$. Then, $t\delta = 0 \in forge(F\delta)$.
   (b) Or $u \neq t$ and u is a pair or an encryption. Then, by the definition of XOR rules one easily verifies
   $$\ulcorner F\delta \urcorner \to_{L_o(t\delta)} \ulcorner F\delta, t\delta. \urcorner$$
   (c) Or $u \neq t$ and u is non standard. In particular, $F \setminus u \to_{L_{oc}(u)} F$ and $u = \text{XOR}(t_1, \ldots, t_n)$ with $t_1, \ldots, t_n \in F \setminus u$. Thus, $F \setminus u \to_{L_o(t)} (F \setminus u), t$ since if u is needed in the construction of t, then the terms $t_1, \ldots, t_n$ can be used. Now, it easily follows that $\ulcorner F\delta \urcorner \to_{L_o(t\delta)} \ulcorner F\delta, t\delta \urcorner$.

□

It remains to show that XOR rules can be applied in polynomial time:

Proposition 3 *Let L_o be the set of XOR rules. Then, the problem whether $E \to_{L_o(t)} E, t$, for a given finite normalized set E of messages and a normalized message t, can be decided in polynomial time with respect to $|E, t|$.*

Proof. Let B be the set of factors of terms in E and S be the factors of t. B and S can be obtained in polynomial time, and it can be decided in polynomial time whether $S \subseteq B$.

- If $S \not\subseteq B$, then t cannot be build from E using an XOR rule.
- Otherwise, $S \subseteq B$. We can represent t by $Factor(t) \subseteq B$. And this set can be represented as a vector of length $|B|$ with entries 0 and 1 where an entry indicates whether a message in B belongs to $Factor(t)$ or not. This vector can be interpreted as an element of the vector space of dimension $|B|$ over the field with two elements. In the same way the terms in E can be represented. Now, deciding $E \to_{L_o(t)} E, t$ is equivalent to deciding whether the vector representing t can be represented as a linear combination of the vectors representing the messages in E. This can be done in polynomial time by gaussian elimination.

□

8.3 Prefix rules

8.3.1 Motivation. As an example, we use the Needham-Schroeder symmetric key authentication protocol [18], which is given as follows:

$$\begin{array}{ll} 1.\ A \to S: & A, B, N_A \\ 2.\ S \to A: & \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}^s_{K_{AS}} \\ 3.\ A \to B: & \{K_{AB}, A\}^s_{K_{BS}} \\ 4.\ B \to A: & \{N_B\}^s_{K_{AB}} \\ 5.\ A \to B: & \{NB-1\}^s_{K_{AB}} \\ 6.\ B \to A: & \{Secret\}_{K_{AB}} \end{array}$$

This protocol is considered to be safe in [7] whereas in [20], a careful analysis of this protocol reveals a flaw in case encryption is carried out by cipher-block-chaining and all atoms are of the size of a block. Our aim is automate such analysis by using deduction rules of the shape:

$$\{\langle M, M'\rangle\}_K \to \{M\}_K.$$

In this example, and using such deduction rules, the intruder can forge $\{N_A, B\}^s_{K_{AS}}$ from the second message, i.e.,

$$\{\langle\langle\langle N_A, B\rangle, K_{AB}\rangle, \{K_{AB}, A\}^s_{K_{BS}}\rangle\}^s_{K_{AS}}.$$

Then, the intruder can send this message to A in another session where B is the initiator of the protocol. In this second session (denoted by $\cdot'$ below), the key N_A accepted by A is also known by the intruder, who can continue the communication with A and derive the secret. More precisely, the attack looks like this:

$$\begin{array}{ll} 1.\ A \to S & : A, B, N_A \\ 2.\ S \to A & : \{N_A, B, K_{AB}, \{K_{AB}, A\}^s_{K_{BS}}\}^s_{K_{AS}} \\ 3'.\ I(B) \to A: & \{N_A, B\}^s_{K_{AS}} \\ 4'.\ A \to I(B): & \{N'_A\}^s_{N_A} \\ 5'.\ I(B) \to A: & \{N'_A\}^s_{N_A} \\ 6'.\ A \to I(B): & \{Secret\}^s_{N_A} \end{array}$$

8.3.2 Prefix rules are oracle rules. We first show that prefix rules allow well formed derivations and then verify the remaining oracle conditions.

Proposition 6 *For all $t \in forge(E)$, there exists a well formed derivation from E with goal g.*

Proof. Let $E_0 = E$ and $D = E_0 \to_{L_1} \ldots \to_{L_n} E_n$ be a derivation of goal g. Let D' be a derivation obtained from D with the following deduction system where the deduction rules are applied with priority order decreasing from 1 to 4.

1. If $i < j$ such that $L_j \in L_{oc}(\{M\}^s_K)$ and $L_i \in L_c(\{\langle .. \langle M, M'_1\rangle \ldots, M'_p\rangle\}^s_K)$, then replace L_j by a sequence of L_d rules decomposing $\langle .. \langle M, M'_1\rangle \ldots, M'_p\rangle$ to M followed by $L_c(\{M\}^s_K)$. Note that the number of L_{oc} rules strictly decreases.

2. If $i < j$ such that $L_i = \{M''\}^s_K \to \{M'\}^s_K \in L_{oc}$ and $L_j = \{M'\}^s_K \to \{M\}^s_K \in L_{oc}$, then replace L_j by the rule $\{M''\}^s_K \to \{M\}^s_K$. Note that the latter rule belongs to L_{oc}. The number of L_{oc} rules does not change but the size of the L_{oc} rule argument strictly increases. (It is bounded by the biggest term in the derivation)

3. If $i < j$ such that $L_i = \{M'\}^s_K \to \{M\}^s_K \in L_{oc}$ and $L_j \in L_d(\{M\}^s_K)$, replace L_j by $L_d(\{M'\}^s_K)$ followed by a sequence of L_d rules decomposing M' to M. The L_{oc} rules do not change but the number of rules $L_d(t)$ such that there exists $L \in D$ with $L \in L_{oc}(t)$ strictly decreases since, due to (2), there exists no $L_{oc}(\{M'\}^s_K)$ rule in D.

4. If there exists $i < n$ such that L_i is a t-rule but, for all $j > i$, L_j does not use t, then remove L_i. (This is removing useless rules)

Clearly, this deduction system terminates: This can easily be shown by defining a (well founded) lexicographical ordering with the different components defined according to the remarks added to the deduction rules above. Then, it is easy to observe that with every application of a deduction rule, the order of a derivation decreases w.r.t. the lexicographical ordering.

It is also clear that the derivation D' derived from D by exhaustively applying the deduction rules and eliminating redundant rules is in fact a derivation from E with goal g. We show that $D' = E'_0 \rightarrow_{L'_1} \ldots \rightarrow_{L'_m} E'_m$ is well formed.

For any rule $L'_i \in L_d(s)$ in D', s is neither obtained with L_c (L'_i would be useless) nor with L_{oc} due to deduction rule (3). Therefore, we have $s \in E$ or there exists $L'_{i<j} \in \mathcal{L}_d(s)$ in D'. By iteration on i, it follows that s is a subterm of E.

For $\mathcal{L}_c$ rules, we will reason by induction on $m - i$. Assume that $L'_i \in \mathcal{L}_c(t)$. If $m - i = 0$, then $t = g$, and therefore, $t \in Sub(E, g)$. For the induction step, there exists a rule L'_j, $j > i$, in D' using t, by the deduction rule (4). If $L'_i \in L_c(t)$, it follows from the definition of derivations that $L'_j \notin L_d(t)$. If $L'_i \in L_{oc}(t)$, we also obtain $L'_j \notin L_d(t)$, by deduction rule (3). Thus, in any case, $L'_j \notin L_d(t)$. By the deduction rules (1) and (2), we conclude $L'_j \notin L_{oc}$. Thus, $L'_j \in L_c(t')$, and t is a subterm of t'. By induction, $t' \in Sub(g, E)$, and thus, $t \in Sub(g, E)$. □

We can now prove, as announced, that these rules are oracle rules:

Proposition 4 *The set L_o of prefix rules is a set of oracle rules.*

Proof. We check each point of the definition:

1. If $t \in forge(E)$, then there exists a well formed derivation from E with goal t, thanks to the Proposition 6.
2. If we have $F \rightarrow_{L_{oc}(\{M\}^s_K)} F, \{M\}^s_K$ using $\{M'\}^s_K$ and $F, \{M\}^s_K \rightarrow_{L_d(\{M\}^s_K)} F, \{M\}^s_K, M$, then as in deduction rule (3) in the proof of Proposition 6 one obtains a derivation from F with goal M without a rule in $L_d(\{M\}_K)$.
3. For any relation $F \rightarrow_{L_{oc}(\{M\}^s_K)} F, \{M\}^s_K$ the proper subterms of $\{M\}^s_K$ are the subterms of M and K, which are also subterms of F.
4. Let u be any non-atomic term. We choose $\epsilon(\langle a, b\rangle) = a$ and $\epsilon(u) = 0$ otherwise. Let F be a set of terms, $0 \in F$, and $\{M\}^s_K$ a term such that $F \setminus \{u\} \rightarrow_L F, u$, with $L \in \mathcal{L}_c(u)$, and $F \rightarrow_{L'} F, \{M\}^s_K$, with $L' \in L_{oc}(\{M\}^s_K)$. Let $\theta = [u \leftarrow \epsilon(u)]$. We distinguish four cases:
   (a) If $u = \{M\}^s_K$, then $\{M\}^s_K\theta = 0 \in forge(F\theta)$.
   (b) Assume that L' uses $u = \{\langle .. \langle M, M'_1\rangle \ldots, M'_p\rangle\}^s_K$. If $L \in L_c(u)$, it follows that $M, K \in F \setminus u$, and thus, $\{M\}^s_K\theta \in forge(F\theta)$. If $L \in L_{oc}(u)$, then $\{\langle .. \langle M, M'_1\rangle \ldots, M'_q\rangle\}^s_K \in F \setminus u$ for some $q > p$ and messages $M'_{p+1}, \ldots, M'_q$. Thus, $\{M\}^s_K\theta \in forge(F\theta)$.
   (c) If L' uses $\{.. \langle M, M'_1\rangle \ldots, M'_p\}^s_K$, $1 \leq q \leq p$ and $u = \langle .. \langle M, M'_1\rangle \ldots, M'_q\rangle$, then $\{M\}^s_K\theta = \{M\}_K \in forge(\{t\}^s_K) \subseteq forge(F\theta)$ with $t = \langle .. \langle M, M'_1\rangle \ldots\rangle, M'_{q-1}\rangle, M'_{q+1}\rangle \ldots, M'_p\rangle$.
   (d) Otherwise, if L' uses $\{t\}^s_K$, then $\{M\}^s_K\theta \in forge(\{t\}^s_K\theta)$.

□

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399