An NP Decision Procedure for Protocol Insecurity with XOR*

Yannick Chevalier†, Ralf Küsters‡, Michaël Rusinowitch†, and Mathieu Turuani†

† LORIA-INRIA-Université Henri Poincaré,
54506 Vandoeuvre-les-Nancy cedex, France
email:{chevalie,rusi,turuani}@loria.fr

‡ Department of Computer Science
Stanford University, Stanford CA 94305, USA
email: kuesters@theory.stanford.edu

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. An immediate consequence of our proof is that checking whether a message can be derived by an intruder (using XOR) is in P. We also apply our framework to an intruder that exploits properties of certain encryption modes such as cipher block chaining (CBC).

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 [14, 18, 22, 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 [21] 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 compromised the others can be compromised 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 protocol is secure.

Recently, several procedures have been proposed to decide insecurity of cryptographic protocols w.r.t. a finite number of protocol sessions [2, 4, 11, 20, 17, 13]. Moreover, some special cases for an unbounded number of sessions have been studied [9, 10, 7, 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 [16, 12], unification algorithms are designed for handling properties of Diffie-Hellman cryptographic systems.

In this paper, we generalize the decidability result of [20], 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.

*This work was partially supported by PROCOPE and IST AVISPA

To the best of our knowledge, the results presented here are the first, besides the ones by Comon and Shmatikov [8] also presented in these proceedings, that go beyond the perfect encryption assumption. We briefly compare our work with [8]: As an immediate consequence of our proof, the problem of checking whether a message can be derived by an intruder in presence of the XOR operator — this problem is called *ground reachability* in [8] — is in PTIME. In [8], this problem is shown to be in NP both for the case of XOR and abelian groups. As for the general insecurity problem, we show NP-completeness based on a theorem that ensures the existence of attacks of linear size. Comon and Shmatikov present a decision procedure with a higher complexity. This procedure is based on constraint solving techniques. However, they consider a more general class of protocol rules. In Section 3.2, we argue that these more general rules are rather unrealistic. Finally, we believe that our framework is quite general in the sense that different intruders with different deduction capabilities can be captured such as those for exploiting properties of encryption based on block ciphers (see Section 7).

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. Proof sketches are provided in Section 5 and 6. Then, in Section 7, XOR rules and prefix rules are introduced and it is shown that these rules are oracle rules, which implies the mentioned complexity results. The missing proofs and an application of prefix rules can be found in [5].

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 [15], i.e., the public-key Needham-Schroeder Procotol with Lowe's fix, where in some place, instead of concatenation XOR is used. Using common notation, the protocol is given as follows:

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

If XOR is interpreted as free symbol, such as pairing, then according to [15] 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 $\mathrm{XOR}(N_A, B, I, B) =_{\mathrm{XOR}} \mathrm{XOR}(N_A, I)$.

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

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, 20, 17] 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:

$$\begin{array}{rcl} term & ::= & \mathcal{A} \mid \mathcal{V} \mid \langle term, term\rangle \\ & & \mid \{term\}^s_{term} \mid \{term\}^p_{\mathcal{K}} \mid \mathrm{XOR}(M) \end{array}$$

where $\mathcal{A}$ is a finite set of constants (*atomic messages*), containing principal names, nonces, keys, and the constants 0 and secret; $\mathcal{K}$ is a subset of $\mathcal{A}$ denoting the set of public and private keys; $\mathcal{V}$ 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 $\mathcal{K}$ 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 $\mathcal{K}$) 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, $\mathcal{V}(t)$ and $\mathcal{V}(E)$ 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 $\mathcal{V}$ 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 $x \in \mathcal{V}$ to $\sigma(x)\delta$.

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

The set of *subterms* of a term t, denoted by $\mathcal{S}(t)$, is defined as follows:

- If $t \in \mathcal{A}$ or $t \in \mathcal{V}$, then $\mathcal{S}(t) = \{t\}$.
- If $t = \langle u, v\rangle$, $\{u\}_v^s$, or $\{u\}_v^p$, then $\mathcal{S}(t) = \{t\} \cup \mathcal{S}(u) \cup \mathcal{S}(v)$.
- If t is non standard, then $\mathcal{S}(t) = \{t\} \cup \bigcup_{u \in \mathcal{F}(t)} \mathcal{S}(u)$.

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

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 *(DAG) size* $|t|$ ($|E|$) of a term t (a set of terms E) is the cardinality of the set $\mathcal{S}(t)$ ($\mathcal{S}(E)$). Note that $|\cdot|$ applied to a set of terms will always denote the DAG size of the set rather than its cardinality.

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* $\ulcorner\_\urcorner : \mathit{term} \to \mathit{term}$. Before providing the formal definition of this function, we illustrate it by some examples: If $a, b, c, d \in \mathcal{A}$, then
- $\ulcorner \text{XOR}(\text{XOR}(a, b, d), \text{XOR}(c, d))\urcorner = \text{XOR}(a, b, c)$
- $\ulcorner \langle \text{XOR}(0, a, a, b, c), \text{XOR}(a, \text{XOR}(a, c))\rangle\urcorner = \langle \text{XOR}(b, c), c\rangle$
- $\ulcorner \text{XOR}(a, \langle \text{XOR}(b), a\rangle, c)\urcorner = \text{XOR}(a, \langle b, a\rangle, c)$.

However, $\ulcorner \text{XOR}(\langle a, b\rangle, \langle a, c\rangle)\urcorner \neq \langle 0, \text{XOR}(b, c)\rangle$.

Formally, the normalization function is recursively defined as follows:

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

$$M(t') := \left(\sum_{t'', \ulcorner t''\urcorner = t'} \mathcal{F}(t)(t'') \right) \bmod 2$$

for every term $t' \neq 0$, and $M(0) := 0$. (Recall that $\mathcal{F}(t)$ is a multiset.) Now, if $M(t') = 0$ for every t', then $\ulcorner t\urcorner := 0$. 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 substitutions are defined. Two terms t and t' are *equivalent* (modulo XOR) if $\ulcorner t\urcorner = \ulcorner t'\urcorner$. In this case, we write $t =_{\text{XOR}} t'$.

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 $(\{R_i \Rightarrow S_i, i \in \mathcal{I}\}, <_\mathcal{I}, E)$ where E is a finite normalized set of messages with $0 \in E$, the* initial intruder knowledge, *$\mathcal{I}$ is a finite (index) set, $<_\mathcal{I}$ is a partial ordering on $\mathcal{I}$, and $R_i \Rightarrow S_i$, for every $i \in \mathcal{I}$, is a protocol rule such that*

1. *the terms R_i and S_i are normalized;*
2. *for all $x \in \mathcal{V}(S_i)$, there exists $j \leq_\mathcal{I} i$ such that $x \in \mathcal{V}(R_j)$;*
3. *for every subterm $\text{XOR}(t_1, \ldots, t_n)$ of R_i, there exists $k \in \{1, \ldots, n\}$ such that $\mathcal{V}(t_l) \subseteq \cup_{j <_\mathcal{I} i} \mathcal{V}(R_j)$ for every $l \in \{1, \ldots, n\} \setminus \{k\}$. (Note that, since R_i is normalized, $t_1, \ldots, t_n$ are standard terms.)*

A bijective mapping $\pi : \mathcal{I}' \to \{1, \ldots, p\}$ is called execution ordering *for P if $\mathcal{I}' \subseteq \mathcal{I}$, p is the cardinality of $\mathcal{I}'$ and for all i, j we have that if $i <_\mathcal{I} j$ and $\pi(j)$ is defined, then $\pi(i)$ is defined and $\pi(i) < \pi(j)$. We define the* size *of π to be p.*

Given a protocol P, in the following we will assume that $\mathcal{A}$ is the set of constants occurring in P. We define $\mathcal{S}(P) := E \cup \bigcup_{i \in \mathcal{I}} (R_i \cup S_i))$ to be the *set of subterms of P*, $|P| := |\mathcal{S}(P)|$ to be the *(DAG) size of P* and $\mathcal{V} := \mathcal{V}(P)$ to be the set of variables occurring in P.

Intuitively, when executing a rule $R_i \Rightarrow S_i$ and on receiving a (normalized) message m in a protocol run, it is

first checked whether m and R_i match, i.e., whether there exists a ground substitution σ such that $m =_{\text{XOR}} R_i\sigma$. If so, $\lceil S_i\sigma \rceil$ 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 $S_i\sigma$ but $\lceil S_i\sigma \rceil$. 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_i and S_i 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_i an output is produced, all variables in S_i 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. We point out that in [8] no restrictions on protocol rules are put, and thus, also these rather unrealistic rules are allowed.

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

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

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 7 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 we have $E \to_L E, t$ (recall that E, t stands for $E \cup \{t\}$) 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 depicted in Table 1. In this table, a, b denote (arbitrary) messages, K is an element of $\mathcal{K}$, and E is a finite set of messages (considered as multiset).

We emphasize that the notion of *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, t 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,
- $L_d := \bigcup_t L_d(t)$, $L_c := \bigcup_t L_c(t)$,
- $L_{od} := \bigcup_t L_{od}(t)$, $L_{oc} := \bigcup_t L_{oc}(t)$,
- $L_o(t) := L_{oc}(t) \cup L_{od}(t)$, $L_o := L_{oc} \cup L_{od}$,
- $\mathcal{L}_d(t)$ is the set of all decomposition t-rules in Table 1, i.e., all t-rule in the left column of the table,
- $\mathcal{L}_d := \bigcup_t \mathcal{L}_d(t)$,
- $\mathcal{L}_c(t)$ is the set of all composition t-rules in Table 1.
- $\mathcal{L}_c := \bigcup_t \mathcal{L}_c(t)$.

- $\mathcal{L} := \mathcal{L}_d \cup \mathcal{L}_c$.

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:

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

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

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, [20].

Definition 2 *Let $P = (\{R'_j \Rightarrow S'_j \mid j \in \mathcal{I}\}, <_{\mathcal{I}}, S_0)$ be a protocol. Then an* attack *on P is a tuple (π, σ) where π is an execution ordering 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 $i \in \{1, \ldots, k\}$ 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 $E \rightarrow_{L_1} E, t_1 \rightarrow_{L_2} \cdots \rightarrow_{L_n} E, t_1, \ldots, t_n$ with a finite set of messages E, messages $t_1, \ldots, t_n$, intruder rules $L_i \in \mathcal{L}$, such that $E, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E, t_1, \ldots, t_i$ and $t_i \notin E \cup \{t_1, \ldots, t_{i-1}\}$, for every $i \in \{1, \ldots, n\}$. The rule L_i is called the *ith rule* in D and the step $E, t_1, \ldots, t_{i-1} \rightarrow_{L_i} E, t_1, \ldots, t_i$ is called the *ith step* in D. We write $L \in D$ to say that $L \in \{L_1, \ldots, L_n\}$. If S is a set of intruder rules, then we write $S \notin D$ to say $S \cap \{L_1, \ldots, L_n\} = \emptyset$. 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 either occurs in the goal or in the initial set of messages.

Definition 3 *Let $D = E \rightarrow_{L_1} \ldots \rightarrow_{L_n} E'$ be a derivation with goal t. Then, D is* well formed *if for every $L \in D$ and t' we have that $L \in \mathcal{L}_c(t')$ implies $t' \in \mathcal{S}(E, t)$, and $L \in \mathcal{L}_d(t')$ implies $t' \in \mathcal{S}(E)$.*

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. They allow 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 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 7, 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.

	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 and such that every proper subterm of a is a subterm of E.

Table 1. Intruder Rules

Theorem 1 *Let L_o be a set of oracle rules. If $E \to t \in^? L_o$ 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. Clearly, the procedure is sound. To show completeness, one has to prove that if there exists an attack (π,σ) on P, then there is one with the size of σ bounded as in step 2. of the procedure. This bound is established in Section 6, Theorem 3. In Section 5, we show that step 3. and 4. in the procedure can be carried out in polynomial time. More precisely, we show that the following problem, henceforth called *derivation problem*, can be solved in polynomial time in the (DAG) size of the input:

$$\text{DERIVE} := \{(E,t) \mid t \in \mathit{forge}(E)\}$$

where E is a finite set of messages and t is a message, both given as DAGs (see Theorem 2). In the procedure, E is the set $\ulcorner\{S_j\sigma \mid j < i\} \cup \{S_0\}\urcorner$ for some $i \in \{1,\ldots k\}$ and t is $\ulcorner R_i\sigma\urcorner$ or secret. From Corollary 1, it follows that $|E,t| \leq 5{\cdot}|P|$, and thus, the procedure depicted in Figure 1 is in fact an NP decision procedure.

5 Deciding the Derivation Problem

We show:

Theorem 2 DERIVE $\in$ *PTIME given that $E \to t \in^? L_o$ can be checked in polynomial time in $|E,t|$ for every finite set E of messages and message t.*

To show this theorem, let $d_t(E)$ be the set consisting of the messages in E and the messages $t' \in \mathcal{S}(E,t)$ that can be derived from E in one step. Using that the number of terms $t' \in \mathcal{S}(E,t)$ is linear in $|E,t|$ and that $E \to t \in^? L_o$ can be checked in polynomial time it is easy to see that $d_t(E)$ can be computed in polynomial time in $|E,t|$. Now, if $t \in \mathit{forge}(E)$, then Definition 4 guarantees that there exists a well formed derivation $D = E \to_{L_1} E, t_1 \to \ldots \to_{L_r} E, t_1, .., t_r$, with $t_r = t$. In particular, $t_i \in \mathit{Sub}(E,t)$ for every $i \in \{1,\ldots,k\}$. By definition of derivations, all t_i are different. It follows $r \leq |t,E|$. Moreover, with $d^0_t(E) := E$ and $d^{l+1}_t(E) := d_t(d^l_t(E))$ we have that $t \in d^{|E,t|}_t(E)$ iff $t \in \mathit{forge}(E)$. Since $d^{|E,t|}_t(E)$ can be computed in polynomial time, Theorem 2 follows.

6. Linear Bounds on Attacks

We now show that the size of an attack can be bounded as required in step 2. in Figure 1.

In what follows, we assume that L_o is a set of oracle rules. If $t \in \mathit{forge}(E)$, we denote by $D_t(E)$ 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.

Definition 5 *Let $P = (\{R_i \Rightarrow S_i, i \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ be a protocol. An attack (π,σ) is* normal *if $|\sigma| := \Sigma_{x\in\mathcal{V}(P)}|\sigma(x)|$ is minimal.*

Clearly, if there is an attack, there is a normal attack. Note, however, normal attacks are not necessarily uniquely determined.

In Lemma 8 we prove, using Lemma 3 to 7, 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 3 and Corollary 1).

Let $P = (\{R_j \Rightarrow S_j, j \in \mathcal{I}\}, <_\mathcal{I}, S_0)$ 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 $i \in \{1,\ldots,k\}$. Recall that $\mathcal{S}(P)$ is the set of subterms of P, $\mathcal{A} \subseteq \mathcal{S}(P)$, and $\mathcal{V} = \mathcal{V}(\mathcal{S}(P))$ is 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 $\ulcorner t\theta\urcorner = t'$.*

Lemma 3 *If (π,σ) is a normal attack, then for all $i \in \{1,\ldots,k\}$, $x \in \mathcal{V}(R_i)$, and standard subterms s of $\sigma(x)$, there exists $j \leq i$ such that $s \in \mathcal{S}(\ulcorner R_j\sigma\urcorner)$ or there exists $t \in \mathcal{S}(P)$ 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 $E \to_L E, t$ then all proper subterms of t are subterms of E.*

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. Check that $\ulcorner R_i\sigma \urcorner \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner)$ for every $i \in \{1, \ldots, k\}$.
4. Check $\mathsf{secret} \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < k+1\} \cup \{S_0\} \urcorner)$.
5. If each check is successful, then answer "yes", and otherwise, "no".

Figure 1. NP Decision Procedure for Insecurity

Proof. For $L \in L_{od} \cup L_{oc}$ use the definition of decomposition and composition guess rules. For $L \in L_d \cup L_c$ the statement is obvious. □

The next lemma states that if a term t' is a subterm of a term t and this term is derived from a set E but t' is not a subterm of E, then t' can be derived from E and the last step of the derivation is a composition rule.

Lemma 5 *Assume that $t' \in \mathcal{S}(t) \backslash \mathcal{S}(E)$ and $t \in \mathit{forge}(E)$, then $t' \in \mathit{forge}(E)$ and there exists a (well formed) derivation from E with goal t' ending with a composition rule.*

Proof. Let $D = E_0 \rightarrow_{L_1} E_1 \cdots \rightarrow_{L_n} E_n$ be a derivation of t from $E_0 = E$. Then, there exists a least $i \neq 0$ such that $t' \in \mathcal{S}(E_i)$ since t' is a subterm of E_n. Assume that L_i is an s-rule for some s. Then, t' is a subterm of s. If t' is a proper subterm of s, Lemma 4 implies that t' is a subterm of E_{i-1} in contradiction to the minimality of i. Thus, $t' = s$ and therefore, $t' \in \mathit{forge}(E)$. By the definition of oracle rules, there exists a well formed derivation D' of t'. If the last step in this derivation is a decomposition rule, then this implies $t' \in \mathcal{S}(E)$ in contradiction to the assumption. Thus, the last step of D' is a composition rule. □

The subsequent lemma will allow us to replace certain subterms occurring in a substitution of an attack by smaller terms. Note that from the assumption made in this lemma it follows that s can be derived from E such that the last rule is a composition rule. This allows to replace s by a smaller term since when deriving t, decomposing s will not be necessary.

Lemma 6 *Let E and F be two sets of normalized messages such that $0 \in E \cup F$. Let $t \in \mathit{forge}(E, F)$ and $s \in \mathit{forge}(E)$ non atomic such that $s \notin \mathcal{S}(E)$. Finally, let δ be the replacement $[s \leftarrow \epsilon(s)]$, where $\epsilon(s)$ is defined as in Definition 4. Then, $\ulcorner t\delta \urcorner \in \mathit{forge}(\ulcorner E\delta, F\delta \urcorner)$.*

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

Lemma 7 *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 8 *Given a normal attack (π, σ), for all variables x and for all factors v_x of $\sigma(x)$, there exists $t \in \mathcal{S}(P)$ such that $t \sqsubseteq_\sigma v_x$.*

Proof. Assume that (*): For every t, $t \sqsubseteq_\sigma v_x$ implies $t \notin \mathcal{S}(P)$. We will lead this to a contradiction. Since $\mathcal{A} \subseteq \mathcal{S}(P)$, we have $v_x \notin \mathcal{A}$, and since v_x is a factor of $\sigma(x)$, v_x is standard. By Lemma 3 and (*), there exists j such that $v_x \in \mathcal{S}(\ulcorner R_j\sigma \urcorner)$. Let N_x be minimal among the possible j. If $v_x \in \mathcal{S}(\ulcorner S_i\sigma \urcorner)$ for some i, (*) implies that there exists $y \in \mathcal{V}(S_i)$ with $v_x \in \mathcal{S}(\sigma(y))$. Then, by Definition 1, (2) there exists $R_{i'}, i' \leq i$ such that $y \in \mathcal{V}(R_{i'})$. Thus, Lemma 3 and (*) imply that there exists $j \leq i$ with $v_x \in \mathcal{S}(\ulcorner R_j\sigma \urcorner)$. Note also that $v_x \notin \mathcal{S}(S_0)$ since otherwise $v_x \in \mathcal{S}(P)$. 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 $\ulcorner R_{N_x}\sigma \urcorner$. Thus, by Lemma 5, $v_x \in \mathit{forge}(E_0)$.

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:

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

We distinguish two cases:

- Assume $j < N_x$. Then, 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 $\ulcorner R_j\sigma \urcorner \in \mathit{forge}(\ulcorner S_0\sigma, \ldots, S_{j-1}\sigma \urcorner)$ it follows $\ulcorner\ulcorner R_j\sigma \urcorner\delta \urcorner \in \mathit{forge}(\ulcorner\ulcorner S_0\sigma \urcorner\delta, \ldots, \ulcorner S_{j-1}\sigma \urcorner\delta \urcorner)$.
- Assume $j \geq N_x$. With $t = \ulcorner R_j\sigma \urcorner$, $s = v_x$, $E = E_0$, and $F = \ulcorner S_{N_x}\sigma, \ldots, S_{j-1}\sigma \urcorner$, Lemma 6 implies $\ulcorner\ulcorner R_j\sigma \urcorner\delta \urcorner \in \mathit{forge}(\ulcorner\ulcorner S_0\sigma \urcorner\delta, \ldots \ulcorner S_{j-1}\sigma \urcorner\delta \urcorner)$.

Thus, $\ulcorner\ulcorner R_j\sigma \urcorner\delta \urcorner \in \mathit{forge}(\ulcorner\ulcorner S_0\sigma \urcorner\delta, \ldots \ulcorner S_{j-1}\sigma \urcorner\delta \urcorner)$ in both cases. Now, with $E = \{S_0, \ldots, S_{j-1}\}$ and $E = \{R_j\}$, respectively, (*) and Lemma 7 imply for all j:

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

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 $|\sigma'| < |\sigma|$, 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 3 *For every protocol P, if (π, σ) is a normal attack on P, then $|\{\sigma(x) \mid x \in \mathcal{V}\}| \leq 4 \cdot |P|$, where $|P|$ is the size of P as defined in Section 3.2.*

Proof. Let $F = \{s \mid \exists x \in \mathcal{V},\ s \in \mathcal{F}(\sigma(x))\}$ 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 $\mathcal{V}' = \{x_s\}_{s \in F}$. The cardinality $\mathsf{Card}(\mathcal{V}')$ of $\mathcal{V}'$ can be bounded as follows:

Claim. $\mathsf{Card}(\mathcal{V}') \leq |P|$
Proof of the claim. We define a function $f : \mathcal{V}' \to \mathcal{S}(P)$ as follows. Due to Lemma 8, for every $y \in \mathcal{V}'$, there exists $t_y \in \mathcal{S}(P)$ such that $t_y \notin \mathcal{V}$ 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, $\mathsf{Card}(\mathcal{V}') \leq |\mathcal{S}(P)| = |P|$, which concludes the proof of the claim.

Let $S = F \cup \{\sigma(x) \mid x \in \mathcal{V}\}$. For all $x \in \mathcal{V}$ let $\sigma''(x) = \ulcorner \mathrm{XOR}(x_{s_1}, \ldots, x_{s_m}) \urcorner$ with $\{s_1, \ldots, s_m\} = \mathcal{F}(\sigma(x))$. 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 $\ulcorner t\varphi \urcorner\sigma' = \ulcorner t\sigma \urcorner$.

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

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

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 $t \in Z_{i+1}$, $v_i \notin \mathcal{F}(t)$. 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 $x \in \mathcal{V}$ with $v = \sigma(x)$ non standard. Then,

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

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

- $v \in F$ and there exists $t \in \mathcal{S}(P)$ 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 $y \in \mathcal{V}(t')$ and $P_{Z_{i+1}} = P_{Z_i} \cup \{t'\}$.

This proves the claim. Using the claim and

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

we obtain

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

□

From this, we obtain:

Corollary 1 *For every protocol P and normal attack (π, σ) on P we have $|R_i\sigma, S_0\sigma, \ldots, S_{i-1}\sigma| \leq 5 \cdot |P|$ and $|\mathsf{secret}, S_0\sigma, \ldots, S_k\sigma| \leq 5 \cdot |P|$, for every $i \in \{1, \ldots, k\}$ with S_j and R_j as defined above.*

7. 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.

7.1. 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 $\{t_1, \ldots, t_n\} \to \ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ with $\{t_1, \ldots, t_n\}$ a non-empty finite multiset of normalized messages such that $\ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ is non-standard, and*
- *L_{od} is the set of rules of the form $\{t_1, \ldots, t_n\} \to \ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ with $\{t_1, \ldots, t_n\}$ a non-empty finite multiset of normalized messages such that $\ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ 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 $\lceil \mathrm{XOR}(t_1, \ldots, t_n) \rceil$ is standard, it is a factor of some of the terms $t_1, \ldots, t_n$. Note that we use that $t_1, \ldots, t_n$ are normalized. Also, the rules in L_{oc} are composition guess rules since proper subterms of $\lceil \mathrm{XOR}(t_1, \ldots, t_n) \rceil$ are subterms of factors of this term, and thus, subterms of $t_1, \ldots, t_n$. Again, we use that $t_1, \ldots, t_n$ are normalized.

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 $s =_{XOR} \mathrm{XOR}(t_1, \ldots, t_n)$ (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), $t \in forge_e(E)$ implies $\lceil t \rceil \in forge(\lceil E \rceil)$.*

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 set of atoms is $\mathcal{A} = \{na, a, I, b, ka, kb, ki, ki^{-1}, 0, \mathsf{secret}\}$ where in Section 2 the secret was N_B. The initial intruder knowledge is $S_0 = \{0, I, ki, ki', ka, kb\}$. 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 $\{\mathsf{secret}, \mathrm{XOR}(na1, I)\}^p_{ka}$ 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 $\sigma(x_{na}) = \mathrm{XOR}(na, b, I)$ and $\sigma(x_{\mathsf{secret}}) = \mathsf{secret}$ is an attack on this protocol. In fact, it is easy to check the steps depicted at the bottom of Figure 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 $E \to t \in L_o(t)$, 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 [20]. Altogether this yields:

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

Together with Proposition 3, Theorem 2 implies:

Theorem 5 *For the XOR intruder, the problem* DERIVE *is in PTIME.*

In [8], this problem is called *ground reachability problem* and is shown to be in NP.

7.2. 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 [5] 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 $\lceil \cdot \rceil$ 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*

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

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, $E \to t \in L_o$ can be decided in polynomial time in $|E, t|$. Also, analogously to the proof in [20] one can show that INSECURE is NP-hard. Now, by Theorem 1, it follows:

Theorem 6 INSECURE *w.r.t. the prefix intruder is an NP-complete problem.*

With Theorem 2 we obtain:

Theorem 7 *For the prefix intruder, the problem* DERIVE *is in PTIME.*

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

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

Figure 2. An attack on the modified NSL protocol

8. Conclusion

We have shown that when extending the standard Dolev-Yao intruder by rules for XORing messages the protocol insecurity problem for a finite number of sessions remains NP-complete. This is the first tight complexity bound given for the insecurity problem without the perfect encryption assumption. Here we have only considered insecurity as failure of secrecy. However, we believe that our result holds also for other properties that can be reduced to reachability problems in our model, such as authentification. 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 *Proc. of CONCUR 2002*, LNCS 2421, pages 499–512. Springer-Verlag, 2002.

[2] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. of CONCUR 2000*, LNCS 1877. Springer-Verlag, 2000.

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

[4] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of ICALP 2001*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.

[5] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. Technical Report RR-4697, INRIA, jan 2003.

[6] Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In *Proc. of CAV'2002*, LNCS 2404, pages 324–337. Springer-Verlag, 2002.

[7] H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping pong protocols. In *Proc. of ICALP'01*, LNCS 2076. Springer-Verlag, 2001.

[8] H. Comon and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *these proceedings*.

[9] D. Dolev and A. Yao. On the security of public key protocols. In *Proc. of FOCS'81*, pages 350–357, 1981.

[10] N. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of Bounded Security Protocols. In *Proc. 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*, 2001.

[12] D. Kapur, P. Narendran, and L. Wang. Analyzing protocols that use modular exponentiation: Semantic unification techniques. In *Proc. of RTA'03*, 2003.

[13] R. Küsters. On the decidability of cryptographic protocols with open-ended data structures. In *Proc. of CONCUR 2002*, LNCS 2421, pages 515–530. Springer-Verlag, 2002.

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

[15] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, LNCS 1055, pages 147–166. Springer-Verlag, 1996.

[16] 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)*, 2002.

[17] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of ACM CCS 2001*, pages 166–175, 2001.

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

[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] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of 14th IEEE Computer Security Foundations Workshop*.

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

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