Available online at www.sciencedirect.com
SCIENCE DIRECT®

Theoretical Computer Science 338 (2005) 247–274

Theoretical Computer Science

www.elsevier.com/locate/tcs

An NP decision procedure for protocol insecurity with XOR☆

Yannick Chevalier[a,1], Ralf Küsters[b], Michaël Rusinowitch[a,*], Mathieu Turuani[a]

[a] *LORIA-INRIA-Université Henri Poincaré, 54506 Vandoeuvre-les-Nancy, France*
[b] *Christian-Albrechts-Universität zu Kiel, Inst. f. Informatik, 24098 Kiel, Germany*

Received 13 November 2003; accepted 11 January 2005

Communicated by M. Wirsing

Abstract

We provide a method for deciding the insecurity of cryptographic protocols in the 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 exclusive or (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 PTIME. We also apply our framework to an intruder that exploits properties of certain encryption modes such as cipher block chaining (CBC).
© 2005 Elsevier B.V. All rights reserved.

Keywords: Verification; Security; XOR; Protocols; Complexity

☆ This work was partially supported by PROCOPE and IST AVISPA. Part of this work was carried out while the second author was at Stanford University supported by the DFG. An extended abstract of this work was published in [6].

* Corresponding author. Tel.: +33 83 593020; fax: +33 83 278319.
E-mail addresses: ychevali@irit.fr (Y. Chevalier), kuesters@ti.informatik.uni-kiel.de (R. Küsters), rusi@loria.fr (M. Rusinowitch), turuani@loria.fr (M. Turuani).

1 Present address: IRIT, 118 route de Narbonne, 31062 Toulouse Cedex 04, France.

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.01.015

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 published security protocols [18,23,29,9,3,4].

While most formal analysis of security protocols abstracts from algebraic properties of operators, 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 is an attack on Bull's recursive authentication protocol first observed by Ryan and Schneider [28]. The protocol is used to distribute a connected chain of keys linking all the nodes from the originator to the server. Ryan and Schneider show that if one key is compromised the others can be compromised too thanks to the property of the exclusive or (XOR). Conversely, if XOR is considered a free operator then, as shown by Paulson using the Isabelle prover [25], 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 [2,5,15,27,22,17]. Moreover, some special cases for an unbounded number of sessions have been studied [13,14,11,1] (see also [20,30] for related work). 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 [21,16], unification algorithms are designed for handling properties of Diffie–Hellman cryptographic systems.

In this paper, we generalize the decidability result of [27], 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) also falls into our framework.

To the best of our knowledge, the decidability results presented here (see also [6]) are the first, besides the ones by Comon and Shmatikov [12], that go beyond the perfect encryption assumption. We briefly compare our work with [12]: we prove that, in the presence of the XOR operator, the problem of checking whether a message can be derived by an intruder is in PTIME—this problem is called *derivation problem* here and *ground reachability problem* in [12]. In [12], the derivation problem is shown to be in NP for the XOR operator and for abelian groups, respectively. 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. Proofs are provided in Sections 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. This section also contains an example illustrating the additional power of prefix rules in attacks.

2. A motivating example

We demonstrate 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 [19], i.e., the public-key Needham–Schroeder Protocol with Lowe's fix, where in some place, instead of concatenation XOR is used. Using common notation (e.g., $\{m\}^p_{K_X}$ denotes the encryption of message m with the public key of agent X), the protocol is given as follows:

1. $A \to B : \{N_A, A\}^p_{K_B}$
2. $B \to A : \{N_B, \text{XOR}(N_A, B)\}^p_{K_A}$
3. $A \to B : \{N_B\}^p_{K_B}$.

If XOR is interpreted as free symbol, such as pairing, then according to [19] 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.

1. $A \to I : \{N_A, A\}^p_{K_I}$,

1$'$. $I(A) \to B : \{\text{XOR}(N_A, B, I), A\}^p_{K_B}$,

2$'$. $B \to I(A) : \{N_B, \text{XOR}(N_A, B, I, B)\}^p_{K_A}$,

2. $I \to A : \{N_B, \text{XOR}(N_A, B, I, B)\}^p_{K_A}$,

3. $A \to I : \{N_B\}^p_{K_I}$.

In step 1′. of this attack, I first decrypts the message $\{N_A, A\}^p_{K_I}$ to obtain N_A and A. Then I applies the XOR operator to compute $\text{XOR}(N_A, B, I)$, before this message together with A is encrypted for B. In 2′ and 2 it is used that the messages $\text{XOR}(N_A, B, I, B)$ and $\text{XOR}(N_A, I)$ are equivalent w.r.t. the properties of XOR, i.e., $\text{XOR}(N_A, B, I, B) =_{\text{XOR}} \text{XOR}(N_A, I)$, and hence, the message in 2 has the form as expected by A. We emphasize that without the intruder's ability to apply the XOR operator and without taking into account algebraic properties of XOR, this attack could not be carried out.

3. The protocol and intruder model

The protocol and intruder model we describe here extends standard models for the (automatic) analysis of security protocols [2,14,27,22] in two respects. First, messages can be built 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 above 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:

$$term ::= \mathcal{A} \mid \mathcal{V} \mid \langle term, term\rangle \mid \{term\}^s_{term} \mid \{term\}^p_{\mathcal{K}} \mid \text{XOR}(M),$$

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 $\text{XOR}(\{t_1, \ldots, t_n\})$ by $\text{XOR}(t_1, \ldots, t_n)$.

Variables are denoted by x, y, terms are denoted by s, t, u, v, and finite sets of terms are written $E, F, \ldots$, 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 $\uplus$ 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 $\text{XOR}(a, b) \notin \mathcal{S}(\text{XOR}(\text{XOR}(a, b), c))$.

We define the size of a term and a set of terms basically as the size of the representation as a *Directed Acyclic Graph* (*DAG*). That is, the (*DAG*) *size* $|t|$ (resp. $|E|$) of a term t (resp. a set of terms E) is the cardinality of the set $\mathcal{S}(t)$ (resp. $\mathcal{S}(E)$). 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* $\ulcorner\_\urcorner$: *term* $\rightarrow$ *term*. Before providing the formal definition of this function, we illustrate it by some examples:

If $a, b, c, d \in \mathcal{A}$, then

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

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:

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

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

  for every term $t' \neq 0$, and $M_t(0) = 0$. (Recall that $\mathcal{F}(t)$ is a multiset.) Now, if $M_t(t') = 0$ for every t', then we set $\ulcorner t\urcorner = 0$. If $M_t(t') \neq 0$ for exactly one t', then we define $\ulcorner t\urcorner = t'$. Otherwise, we set $\ulcorner t\urcorner = \text{XOR}(M_t)$.

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 $t =_{\mathrm{XOR}} t'$.

One easily shows:

Lemma 1. *For every $n \geqslant 0$, term t, and substitution σ:*
(1) $|\ulcorner t \urcorner| \leqslant |t|$, *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 2. For every normalized term t with $|t| \leqslant n$, 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 of protocol rules, protocols, and execution orderings is explained below.

Definition 3. 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 $j \leqslant_{\mathcal{I}} i$ such that $x \in \mathcal{V}(R_j)$;
(3) for every subterm $\mathrm{XOR}(t_1, \ldots, t_n)$ 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 $\pi(i) < \pi(j)$. Let p be the *size* of π.

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 $m =_{\mathrm{XOR}} R_i\sigma$. If so, $\ulcorner S_i\sigma \urcorner$ 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 $S_i\sigma$ but $\ulcorner S_i\sigma \urcorner$. 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_i and S_i may be binded 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.

Before we explain Conditions 1–3 in Definition 3, we formalize the protocol informally described in Section 2: 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 $S_0 = \{0, I, ki, ki^{-1}, ka, kb\}$. The protocol rules are the following, where we have only retained the rules that are used in the attack:

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

Obviously, Conditions 1–3 are satisfied. We have $\mathcal{I} = \{(a,1), (a,2), (b,1)\}$ and $<_{\mathcal{I}} = \{((a,1), (a,2))\}$.

We now discuss the three conditions of Definition 3. Condition 1 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, the substitutions of all variables in S_i are determined already. Otherwise, the output of a protocol rule would be arbitrary, which is of course undesirable. For instance, if a protocol contains a protocol rule of the form $a \Rightarrow x$ where the variable x occurs for the first time, then x can be substituted by an arbitrary message, and hence, arbitrary output can be produced by this rule. By Condition 2 such undesirable phenomena are excluded. Condition 3 guarantees that when applying a protocol rule, the substitution of the variables in this rule can uniquely and immediately be determined when given the input from the intruder for this rule. For example, if a protocol contains a rule of the form $\textsc{xor}(x, y) \Rightarrow \langle x, y\rangle$ where the variables x and y occur for the first time, then this protocol violates Condition 3: on receiving $\textsc{xor}(a, b, c)$, for instance, infinitely many substitutions are possible, including $\{x \mapsto \textsc{xor}(a, b), y \mapsto c\}$, $\{x \mapsto \textsc{xor}(b, \langle a, a\rangle), y \mapsto \textsc{xor}(a, c, \langle a, a\rangle)\}$, $\{x \mapsto \textsc{xor}(b, \langle a, \langle a, a\rangle\rangle), y \mapsto \textsc{xor}(a, c, \langle a, \langle a, a\rangle\rangle)\}$ etc. In other words, a principal can arbitrarily pick a substitution out of infinitely many possible substitutions. With Condition 3 we avoid this. The following two examples illustrate that when Condition 3 is not satisfied, then this is due to an “unreasonable” specification of the protocol.

We start with a very simple protocol which contains two rules, (A,i) and (A,j), where (A,i) preceeds (A,j). Intuitively, these rules are part of a description of a principal A who would first apply (A, i) before performing (A, j)

$$\begin{array}{lrcl}
(A,i): & \textsc{xor}(x, y) & \Rightarrow & \cdots \\
 & & \vdots & \\
(A,j): & x & \Rightarrow & y
\end{array}$$

Informally speaking, in (A, i) principal A accepts a message which A believes to be of the form $\textsc{xor}(x, y)$. However, since neither x nor y is known at this point, A has to accept any message. Only when performing (A, j) one part of the message, namely x, is determined, and from this together with $\text{XOR}(x, y)$ A can compute the second part, y. The problem with the above description is that (A, i) cannot be performed deterministically. Principal A has to guess some x and y (infinitely many substitutions are possible). Only later if (A, j) is performed x and y can be determined. However, in an attack (A, j) might never be applied. This phenomenon will occur in every protocol where Condition 3 is not satisfied and it is due to an ill-defined protocol specification. In the example, the way the protocol should be modeled is the following:

$$\begin{array}{ll} (A, i): & z \Rightarrow \cdots \\ & \quad\vdots \\ (A, j): & x \Rightarrow \text{XOR}(z, y) \end{array}$$

Obviously, in this formulation of the protocol Condition 3 is satisfied.

The second example presents a protocol where a message is received at one point of the protocol run and decrypted at a subsequent step. In the informal Alice and Bob notation, the protocol is stated as follows:

$$\begin{array}{lc} 1.\ A \to B: & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K), \\ 2.\ B \to A: & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K), B), \\ 3.\ A \to B: & K, \\ 4.\ B \to A: & N_a. \end{array}$$

In this protocol, principal B receives $\{N_a\}_K \oplus \{B\}_K$ from A and returns $\{N_a\}^s_K \oplus \{B\}^s_K \oplus B$. When A sends the key K, B is able to retrieve N_a from the message previously received from A. The following is a naïve formalization of this protocol where the partial ordering of the rules is defined in the obvious way:

$$\begin{array}{lccc} (A, 1): & 0 & \Rightarrow & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K), \\ (B, 1): & \text{XOR}(\{x\}_y, \{B\}_y) & \Rightarrow & \text{XOR}(\{x\}_y, \{B\}_y, B), \\ (A, 2): & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K, B) & \Rightarrow & K, \\ (B, 2): & y & \Rightarrow & x. \end{array}$$

First note that because of $(B, 1)$, Condition 3 is not satisfied. However, this formulation of the protocol is unrealistic because in $(B, 1)$ principal B only accepts messages which are obtained as the XOR of two encrypted messages whose key and in case of the first encrypted message whose plaintext B does not know. Hence, in a realistic implementation B is not able to check whether the received message has the required form, but rather has to accept any message. The protocol should therefore be stated as follows:

$$\begin{array}{lccc} (A, 1): & 0 & \Rightarrow & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K), \\ (B, 1): & z & \Rightarrow & \text{XOR}(z, B), \\ (A, 2): & \text{XOR}(\{N_a\}^s_K, \{B\}^s_K, B) & \Rightarrow & K, \\ (B, 2): & y & \Rightarrow & \{z\}^s_{Nb}, \\ (B, 3): & \text{XOR}(\{\{x\}^s_y, \{B\}^s_y\}^s_{Nb}) & \Rightarrow & x. \end{array}$$

where N_b is a fresh nonce which is only known to B. In steps $(B, 2)$ and $(B, 3)$ we describe that B receives a message y, supposingly K, which is then used to extract N_a from (the message substituted for) z. Note that this formulation of the protocol satisfies Condition 3.

We point out that in [12] no restrictions on protocol rules are put, and thus, as illustrated above, also "unreasonable" protocol specifications, where Condition 3 is not satisfied, are allowed.

3.3. *The intruder model and attacks*

Our intruder model follows the Dolev–Yao intruder [13]. 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, the 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 Fig. 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 Fig. 1. For now, there may be any set of guess rules of the kind shown in Fig. 1, later we will consider certain classes of guess rules, namely oracle rules.

We refer to certain sets of intruder rules using the notation as depicted in Fig. 1. For example, $L_{p1}(\langle a, b\rangle)$ refers to the singleton set $\{\langle a, b\rangle \to a\}$ and $L_c(\langle a, b\rangle)$ to the singleton set $\{a, b \to \langle a, b\rangle\}$. In the same way $L_{p2}(\langle a, b\rangle)$, $L_{ad}(\{a\}^p_K)$, $L_{sd}(\{a\}^s_b)$, $L_c(\{a\}^p_K)$, and $L_c(\{a\}^s_b)$ define sets consisting of one intruder rule. With $L_{od}(a)$ and $L_{oc}(a)$ we denote (finite or infinite) sets of guess rules. Note that, even if no guess rules are considered, i.e., for every message a the sets $L_{od}(a)$ and $L_{oc}(a)$ are empty, the number of decomposition and composition rules is always infinite since there are infinitely many messages a, b. The reason $L_{p1}(\langle a, b\rangle)$, $L_{p2}(\langle a, b\rangle)$, $L_{ad}(\{a\}^p_K)$, $L_{sd}(\{a\}^s_b)$, $L_c(\langle a, b\rangle)$, $L_c(\{a\}^p_K)$, and $L_c(\{a\}^s_b)$ denote singletons rather than intruder rules is simply notational convenience.

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,
- $L_d = \bigcup_t L_d(t)$, $L_c = \bigcup_t L_c(t)$,
- $L_{od} = \bigcup_t L_{od}(t)$, $L_{oc} = \bigcup_t L_{oc}(t)$,

	Decomposition rules	Composition rules
Pair	$L_{p1}(\langle a,b\rangle)$: $\langle a,b\rangle \to a$ $L_{p2}(\langle a,b\rangle)$: $\langle a,b\rangle \to b$	$L_c(\langle a,b\rangle)$: $a, b \to \langle a,b\rangle$
Asym.	$L_{ad}(\{a\}_K^p)$: $\{a\}_K^p, K^{-1} \to a$	$L_c(\{a\}_K^p)$: $a, K \to \{a\}_K^p$
Sym.	$L_{sd}(\{a\}_b^s)$: $\{a\}_b^s, b \to a$	$L_c(\{a\}_b^s)$: $a, b \to \{a\}_b^s$
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.

Fig. 1. Intruder rules.

- $L_o(t) = L_{oc}(t) \cup L_{od}(t)$, $L_o = L_{oc} \cup L_{od}$,
- $\mathcal{L}_d(t)$ is the set of all decomposition t-rules in Fig. 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 Fig. 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

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

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

Lemma 4. *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, [27].

Definition 5. 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 $\ulcorner R_i\sigma \urcorner \in \mathit{forge}(\ulcorner S_0, S_1\sigma, \dots, S_{i-1}\sigma\urcorner)$ 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 $\mathsf{secret} \in \mathit{forge}(\ulcorner S_0, S_1\sigma, \dots, S_k\sigma\urcorner)$.

Due to Lemma 1, it does not matter whether, in the above definition, σ is normalized or not. Also note that Lemma 4 implies: $\ulcorner \mathit{forge}(\ulcorner S_0, S_1\sigma, \dots, S_{i-1}\sigma\urcorner)\urcorner = \mathit{forge}(\ulcorner S_0, S_1\sigma, \dots, S_{i-1}\sigma\urcorner)$.

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

$$\textsc{Insecure} = \{P \mid \text{ there exists an attack on} P\}.$$

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, $n \geqslant 0$, 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 either occurs in the goal or in the initial set of messages.

Definition 6. 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 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. Condition 2 says that to derive a from E it is not necessary to first compose a message t from E using an oracle rule and then decompose t to obtain a. Condition 3 allows us to replace a message u which can be composed from $F \setminus u$ by a smaller message $\varepsilon(u)$ in an application of an oracle rule. Conditions 2 and 3 together are later used (see Section 6) to bound the size of the substitution σ of an attack. They allow us to replace a subterm u in σ, composed by the intruder, by a smaller message.

Definition 7. 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 $t \in \mathit{forge}(E)$, then there exists a well-formed derivation from E with goal t.

```
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 $\pi$ for $P$. Let $k$ be the size of $\pi$.
    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 $\sigma$ such that $|\{\sigma(x) \mid x \in V\}| \leqslant 3n$.
(3) Check that $\ulcorner R_i\sigma \urcorner \in forge(\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner)$ for every $i \in \{1, \ldots, k\}$.
(4) Check secret $\in forge(\ulcorner \{S_j\sigma \mid j < k+1\} \cup \{S_0\} \urcorner)$.
(5) If each check is successful, then answer "yes", and otherwise, "no".
```

Fig. 2. NP decision procedure for insecurity.

(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 $\varepsilon(u)$ with $|\varepsilon(u)| < |\ulcorner u \urcorner|$ 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 $\ulcorner t[u \leftarrow \varepsilon(u)] \urcorner \in forge(\ulcorner F[u \leftarrow \varepsilon(u)] \urcorner)$ and $\varepsilon(u) \in forge(F)$ for every message t.

In Section 7 we will present sets of oracle rules. An example for guess rules which do not form a set of oracle rules is the following: $L_{od}(t) = \{\{t\}^s_c \rightarrow t\}$ and $L_{oc}(\{t\}^s_{\langle a,b\rangle}) = \{\{t\}^s_{\langle a,b\rangle} \rightarrow \{t\}^s_c\}$ where t is an arbitrary message and a, b, c are fixed atomic messages. That is, $L_{oc}(\{t\}^s_{\langle a,b\rangle})$ allows to turn an encryption with key $\langle a, b\rangle$ into an encryption with key c. Using $L_{od}(t)$ such a message can then be decrypted. It is easy to check that neither of the three conditions in Definition 7 is satisfied.

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 the presence of an intruder that uses XOR rules and prefix rules, respectively.

Theorem 8. *Let L_o be a set of oracle rules. If $E \rightarrow t \in^? L_o$ 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 Fig. 2. In the following two sections, we show that this procedure is sound and complete, and that it runs in non-deterministic polynomial time. From this, Theorem 8 follows immediately.

Clearly, the procedure is sound. In Section 5 we show that the procedure runs in non-deterministic polynomial time. To this end, we prove that the derivation problem (called ground reachability problem in [12]) can be decided in polynomial time (Theorem 9). This result is of independent interest. As a corollary, we obtain the desired complexity bound (Corollary 10). Completeness of our procedure is then established in Section 6 where we show 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 (Theorem 20).

5. Deciding the derivation problem

The derivation problem is defined as follows:

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

where E is a finite set of messages and t is a message, both given as DAGs.

We show:

Theorem 9. 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.*

Proof. 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 7 guarantees that there exists a well-formed derivation $D = E \to_{L_1} E, t_1 \to \cdots \to_{L_r} E, t_1, \ldots, t_r$, 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 $r \leqslant |t, E|$. Moreover, with $d_t^0(E) = E$ and $d_t^{l+1}(E) = d_t(d_t^l(E))$ we have that $t \in d_t^{|E,t|}(E)$ iff $t \in \mathit{forge}(E)$. Since $d_t^{|E,t|}(E)$ can be computed in polynomial time, Theorem 9 follows. $\Box$

As an immediate consequence we obtain:

Corollary 10. *The procedure depicted in Fig.* 2 *runs in polynomial time in* $|P|$ *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.*

Proof. It suffices to observe that steps 3 and 4 of our procedure can be performed in deterministic polynomial time in $|P|$. Given σ with $|\{\sigma(x) \mid x \in V\}| \leqslant 3n$ and using Lemma 1 we have that $|\ulcorner R_i\sigma, S_0\sigma, \ldots, S_{i-1}\sigma\urcorner| \leqslant |R_i\sigma, S_0\sigma, \ldots, S_{i-1}\sigma| \leqslant |R_i, S_0, \ldots, S_{i-1}, \sigma(\mathcal{V})| \leqslant |P| + 3 \cdot |P| \leqslant 4 \cdot |P|$. Similarly for $\left|\ulcorner \mathsf{secret}, S_0\sigma, \ldots, S_k\sigma\urcorner\right|$. Hence, using Theorem 9 it follows that steps 3–4 of our procedure can be performed in deterministic polynomial time in $|P|$. $\Box$

6. Linear bounds on attacks

We now show that the size of an attack can be bounded as required in step 2 of the algorithm depicted in Fig. 2.

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 11. 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, that normal attacks are not necessarily uniquely determined.

In Lemma 19 we prove, using Lemmas 13–18, 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 20 and Corollary 10).

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 $i \in \{1, \dots, k\}$. 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 12. 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 following lemma says that standard subterms s occurring in the substitution σ of a normal attack start with a subterm of the protocol under consideration or (at least) occur on the left-hand side of a protocol rule when the substitution is applied. Note that even if s occurs in $\sigma(x)$ and x occurs in R_i, s does not need to occur in $\ulcorner R_i \sigma \urcorner$ because of the normalization.

Lemma 13. *If (π, σ) is a normal attack, then for all $i \in \{1, \dots, k\}$, $x \in \mathcal{V}(R_i)$, and standard subterms s of $\sigma(x)$, there exists $j \leqslant i$ such that $s \in \mathcal{S}(\ulcorner R_j \sigma \urcorner)$ or there exists $t \in \mathcal{S}(P)$ with $t \sqsubseteq_\sigma s$.*

Proof. Assume that there exists $i \in \{1, \dots, k\}$, $x \in \mathcal{V}(R_i)$, and s standard subterm of $\sigma(x)$ such that for all $j \leqslant i$: $s \notin \mathcal{S}(\ulcorner R_j \sigma \urcorner)$. Define

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

and let y be a variable of R_j such that s is a subterm of $\sigma(y)$. Note that $j \leqslant i$, and thus, $s \notin \mathcal{S}(\ulcorner R_j \sigma \urcorner)$. Let $S_{y,s}$ be the set of subterms t of R_j such that $y \in \mathcal{V}(t)$ 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 $r \in \mathcal{S}(R_j)$ with $t \in \mathcal{S}(r)$ and there exists no $r' \in \mathcal{S}(r)$ with $t \in \mathcal{S}(r')$. Then, since $s \notin \mathcal{S}(\ulcorner r\sigma \urcorner)$, r must be of the form $\text{XOR}(t, t_1, \dots, t_n)$ with $t, t_1, \dots, t_n$ standard (since R_j is normalized) and $n \geqslant 1$ such that there exists $i \in \{1, \dots, n\}$, say $i = 1$, with $\ulcorner t\sigma \urcorner = \ulcorner t_1 \sigma \urcorner$ (s has been eliminated by normalization). In particular, $s \in \mathcal{S}(\ulcorner t_1 \sigma \urcorner)$. (This situation is depicted in Fig. 3.)

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

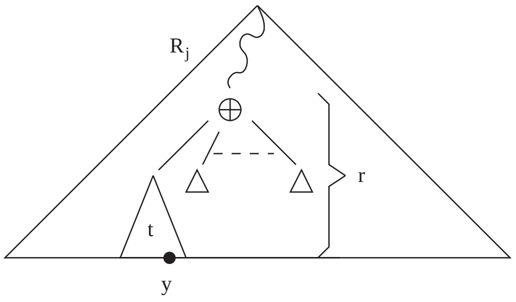

Fig. 3. Structure of R_j.

Roughly speaking, the following Lemma 14 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 $D_1 = E_1 \to \cdots \to F_1$ and $D_2 = E_2 \to \cdots \to F_2$ 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 14. *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 $D'' = D.D_t(E) = D.D''' - D'''$ is obtained from $D_t(E)$ 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 7, (2) implies that every step in D''' of the form $F_1, F_2, \gamma \to_{L_d(\gamma)} F_1, F_2, \gamma, \beta$ 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. □

The proof of the following lemma is trivial.

Lemma 15. *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.*

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 16. *Assume that $t' \in \mathcal{S}(t) \setminus \mathcal{S}(E)$ and $t \in forge(E)$, then $t' \in forge(E)$ 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 15 implies that t' is a subterm of E_{i-1} in contradiction to the minimality of i. Thus, $t' = s$ and therefore, $t' \in forge(E)$. 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. $\square$

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 17. *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 $s \notin \mathcal{S}(E)$. Finally, let δ be the replacement $[s \leftarrow \varepsilon(s)]$, where $\varepsilon(s)$ is defined as in Definition 7. Then, $\ulcorner t\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.*

Proof. Let $D_s = E \rightarrow_{L_1} E, t_1 \rightarrow_{L_2} \cdots \rightarrow_{L_p} E, t_1, \ldots, t_p \rightarrow_{\mathcal{L}_c(s)} E, t_1, \ldots, t_p, s$. By induction on i and using Lemma 15, it follows that all proper subterms of t_i are subterms of $E, t_1, \ldots, t_{i-1}$. Using $s \notin \mathcal{S}(E)$ and $s \neq t_i$, this implies $s \notin \mathcal{S}(t_i)$, 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 14, there exists a derivation $D = E, F, t_1, \ldots, t_p \rightarrow_{L_{p+1}} E, F, t_1, \ldots, t_{p+1} \rightarrow \cdots \rightarrow_{L_n} E, F, t_1, \ldots, t_n$ 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, $p \leqslant i \leqslant n$, that $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$. For $i = p$ this is by Definition 7, (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 $\varepsilon(s)$, $\ulcorner t_i\delta \urcorner = \varepsilon(s) \in forge(\ulcorner E\delta, F\delta \urcorner)$, or $\ulcorner t_i\delta \urcorner = \left\langle \ulcorner a\delta \urcorner, \ulcorner b\delta \urcorner \right\rangle \in forge(\ulcorner E\delta, F\delta \urcorner)$ since $\{a, b\} \subseteq E \cup F \cup \{t_1, \ldots, t_{i-1}\}$. 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 $L_i \notin L_d(s)$, and 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 $\langle t_i, a\rangle \in E \cup F \cup \{t_1, \ldots, t_{i-1}\}$. Analogously for L_{p2}, L_{sd}, and L_{ad}.
- If $L_i \in L_{oc} \cup L_{od}$, then thanks to Definition 7, (4), we have: $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta, t_1\delta, \ldots, t_{i-1}\delta \urcorner)$ and thus: $\ulcorner t_i\delta \urcorner \in forge(\ulcorner E\delta, F\delta \urcorner)$.

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

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

Lemma 18. *Let σ be a normalized ground substitution, E a set of normalized terms, s a normalized standard non-atomic term, and δ the replacement $[s \leftarrow \varepsilon(s)]$. 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 $t \in \mathcal{S}(E)$, we have $\ulcorner t\sigma' \urcorner = \ulcorner \ulcorner t\sigma \urcorner \delta \urcorner$. This will conclude the proof of the lemma.

- If $t \in \mathcal{A}$, then $t \neq s$ by assumption. Thus, $\ulcorner \ulcorner t\sigma \urcorner \delta \urcorner = t = \ulcorner t\sigma' \urcorner$.
- If $t \in \mathcal{V}$, 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 $s \neq \left\langle \ulcorner v\sigma \urcorner, \ulcorner w\sigma \urcorner \right\rangle$ since otherwise $t \sqsubseteq_\sigma s$, and $\ulcorner t\sigma' \urcorner = \left\langle \ulcorner v\sigma' \urcorner, \ulcorner w\sigma' \urcorner \right\rangle$. By induction, this gives $\ulcorner t\sigma' \urcorner = \left\langle \ulcorner \ulcorner v\sigma \urcorner \delta \urcorner, \ulcorner \ulcorner w\sigma \urcorner \delta \urcorner \right\rangle$, 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 $t = \{u\}_v^s$ and $t = \{u\}_K^p$ are similar.
- If $t = \textsc{xor}(\mathcal{T})$, where $\mathcal{T}$ is a multiset of standard terms, we have:

$$\begin{aligned}
\ulcorner t\sigma' \urcorner &= \ulcorner \textsc{xor}(\{t'\sigma' \mid t' \in \mathcal{T}\}) \urcorner \\
&= \ulcorner \textsc{xor}(\{\ulcorner t'\sigma' \urcorner \mid t' \in \mathcal{T}\}) \urcorner && \text{(Definition of } \ulcorner\_\urcorner\text{)} \\
&= \ulcorner \textsc{xor}(\{\ulcorner \ulcorner t'\sigma \urcorner \delta \urcorner \mid t' \in \mathcal{T}\}) \urcorner && \text{(By induction on every } t' \in \mathcal{T}\text{)} \\
&= \ulcorner \textsc{xor}(\{\ulcorner t'\sigma \urcorner \delta \mid t' \in \mathcal{T}\}) \urcorner && \text{(Definition of } \ulcorner\_\urcorner\text{)} \\
&= \ulcorner \textsc{xor}(\{\ulcorner t'\sigma \urcorner \mid t' \in \mathcal{T}\})\delta \urcorner && \text{(Definition of } \delta \text{ and } s \text{ standard)} \\
&= \ulcorner \ulcorner t\sigma \urcorner \delta \urcorner && \text{(Definition of } \ulcorner\_\urcorner\text{).} \qquad \square
\end{aligned}$$

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 19. *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 13 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 3, (2) there exists $R_{i'}$, $i' \leqslant i$ such that $y \in \mathcal{V}(R_{i'})$. Thus, Lemma 13 and (*) imply that there exists $j \leqslant i$ 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 $i \geqslant N_x$. 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 16, $v_x \in \mathit{forge}(E_0)$.

Let us define the replacement $\delta = [v_x \leftarrow \varepsilon(v_x)]$ where $\varepsilon(v_x)$ is defined as in Definition 7. 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 first $j < N_x$. Then, by minimality of N_x, v_x is neither a subterm of $\ulcorner R_j\sigma \urcorner$ nor a subterm of $\ulcorner S_0\sigma, \ldots, S_{j-1}\sigma \urcorner$. 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 now that $j \geqslant N_x$. 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 17 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, in both cases

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

Now, with $E = \{S_0, \ldots, S_{j-1}\}$ and $E = \{R_j\}$, respectively, (*) and Lemma 18 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 $\varepsilon(v_x)$, 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 20. *For every protocol P, if* (π, σ) *is a normal attack on P, then* $|\{\sigma(x) \mid x \in \mathcal{V}\}| \leqslant 3 \cdot |P|$, *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. $\mathsf{Card}(\mathcal{V}') \leqslant |P|$.

Proof of the claim. We define a function $f : \mathcal{V}' \to \mathcal{S}(P)$ as follows. Due to Lemma 19, 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, $\mathsf{Card}(\mathcal{V}') \leqslant |\mathcal{S}(P)| = |P|$, 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, $\sigma(x) = \sigma'(\sigma''(x))$.

$$\ulcorner\ulcorner t\sigma''\urcorner\sigma'\urcorner = \ulcorner t\sigma\urcorner \quad \text{for all } t.$$

Hence $\ulcorner t\sigma''\urcorner$ is a σ'-match of $\ulcorner t\sigma\urcorner$. However $\ulcorner t\sigma''\urcorner\sigma'$ is not necessarily normalized which might be problematic for the sequel. Hence let us build another term $\ulcorner t\sigma''\urcorner^{\sigma'}$ from $\ulcorner t\sigma''\urcorner$, such that $\ulcorner t\sigma''\urcorner^{\sigma'}\sigma' = \ulcorner t\sigma\urcorner$. Intuitively, it amounts to eliminate all subterms that "would get deleted" in $\ulcorner t\sigma''\urcorner$ by normalizing $\ulcorner t\sigma''\urcorner\sigma'$. Let us define:

$$\mathrm{XOR}(v_1, \ldots, v_n)^{\sigma'} = \begin{cases} \mathrm{XOR}(v_1, \ldots, v_n \backslash v_i \backslash v_j)^{\sigma'} & \text{if } \ulcorner v_i\sigma'\urcorner = \ulcorner v_j\sigma'\urcorner \text{ and } i \neq j \\ \mathrm{XOR}(v_1^{\sigma'}, \ldots, v_n^{\sigma'}) & \text{otherwise.} \end{cases}$$

$$\langle a, b\rangle^{\sigma'} = \langle a^{\sigma'}, b^{\sigma'}\rangle \text{ and similarly for } \{a\}_b^p \text{ and } \{a\}_b^s.$$

$$a^{\sigma'} = a \qquad\qquad \text{if } a \in Var \cup Var' \cup \mathit{Atoms},$$

where $\mathrm{XOR}(v_1, \ldots, v_n \backslash v_i \backslash v_j)$ represents the XOR of all v_i for $i \in \{1, \ldots, n\}\backslash\{i, j\}$. We can check that $\ulcorner\ulcorner t\sigma''\urcorner^{\sigma'}\sigma'\urcorner = \ulcorner t\sigma\urcorner$ for all *t*, since by the above transformation we have $\ulcorner \mathrm{XOR}(v_1, \ldots, v_n)^{\sigma'}\sigma'\urcorner = \ulcorner \mathrm{XOR}(v_1, \ldots, v_n)\sigma'\urcorner$. Moreover for all $\mathrm{XOR}(u_1, \ldots, u_k)$ subterm of $\ulcorner t\sigma''\urcorner^{\sigma'}$, we have $u_i\sigma'$ standard (since $\ulcorner t\sigma''\urcorner$ is normalized and $\sigma'(x_s)$ is standard, for all

$s \in F$), and it follows[2] that $\ulcorner u_i\sigma' \urcorner \neq 0$, and $\ulcorner u_i\sigma' \urcorner \neq \ulcorner u_j\sigma' \urcorner$ for all $i \neq j$, by definition of $\ulcorner t\sigma' \urcorner^{\sigma'}$. By consequence $\ulcorner t\sigma' \urcorner^{\sigma'}\sigma'$ is normalized and then

$$\ulcorner t\sigma' \urcorner^{\sigma'}\sigma' = \ulcorner t\sigma \urcorner \quad \text{for all terms } t.$$

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

$$V_Z = \{x \in \mathcal{V} \mid \sigma(x) \text{ non-standard and } \sigma(x) \notin Z\},$$
$$P_Z = \{\ulcorner t\sigma' \urcorner^{\sigma'} \mid t \in \mathcal{S}(P) \text{ and } \ulcorner t\sigma \urcorner \notin Z\}.$$

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

Claim. $|S \cup P_S| \leqslant |V_\emptyset \cup P_\emptyset|$.

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

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

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,

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

  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 $t' = \ulcorner t\sigma“ \urcorner^{\sigma'}$. We have $t'\sigma' = \ulcorner t\sigma \urcorner = v$. Then,

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

  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

$$|P_\emptyset| = \left|\ulcorner \mathcal{S}(P)\mathcal{V}' \urcorner\right| \leqslant \left|\mathcal{S}(P)\mathcal{V}'\right| \leqslant |\mathcal{S}(P)| + \left|\mathcal{V}'\right|,$$

we obtain

$$\begin{aligned} |\{\sigma(x) \mid x \in \mathcal{V}\}| &\leqslant |S| \leqslant |S \cup P_S| \leqslant |V_\emptyset \cup P_\emptyset| \\ &\leqslant |\mathcal{V}| + |\mathcal{S}(P)| + |\mathcal{V}'| \leqslant 3 \cdot |P| \qquad \square \end{aligned}$$

From this theorem completeness of the procedure depicted in Fig. 2 follows immediately.

[2] Remark that $\sigma'(x) \neq 0$ for all x.

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 21. 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 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 $\ulcorner \text{XOR}(t_1, \ldots, t_n) \urcorner$ 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 $s =_{XOR} \text{XOR}(t_1, \ldots, t_n)$ (not necessarily normalized). Let $\mathit{forge}_e(E)$ 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 22. *For every message term t and set of messages E* (*both not necessarily normalized*), *$t \in \mathit{forge}_e(E)$ implies $\ulcorner t \urcorner \in \mathit{forge}(\ulcorner E \urcorner)$.*

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

7.1.1. Example

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

We recall (see Section 3.2) that the protocol underlying the attack is formally described as follows: the initial intruder knowledge is $\{0, I, ki, ki^{-1}, ka, kb\} = S_0$. The protocol rules are

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

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 $\{\mathsf{secret}, \text{XOR}(na, I)\}^p_{ka}$ 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 $\sigma(x_{\mathsf{secret}}) = \mathsf{secret}$ is an attack on this protocol. In fact, it is easy to check that the three following statements are true:

$$\begin{array}{l}\{\langle \text{XOR}(na, b, I), a\rangle\}^p_{kb} \in \mathit{forge}(0, I, ki, ki^{-1}, ka, kb, \{\langle na, a\rangle\}^p_{ki}),\\ \{\langle \mathsf{secret}, \text{XOR}(na, I)\rangle\}^p_{ka} \in \mathit{forge}(0, I, ki, ki^{-1}, ka, kb, \{\langle na, a\rangle\}^p_{ki}, \mathit{msg})\\ \qquad\qquad \text{with } \mathit{msg} = \ulcorner\{\langle \mathsf{secret}, \text{XOR}(\text{XOR}(na, b, I), b)\rangle\}^p_{ka}\urcorner,\\ \mathsf{secret} \in \mathit{forge}(0, I, ki, ki^{-1}, ka, kb, \{\langle na, a\rangle\}^p_{ki}, msg, \{\mathsf{secret}\}^p_{ki})\\ \qquad\qquad \text{with } \mathit{msg} = \ulcorner\{\langle \mathsf{secret}, \text{XOR}(\text{XOR}(na, b, I), b)\rangle\}^p_{ka}\urcorner.\end{array}$$

7.1.2. XOR rules are oracle rules

We now show that the XOR rules form a set of oracle rules. We start to show Definition 7, (1). To do so, we first prove a sufficient condition for a derivation to be well-formed.

Lemma 23. *Let $D = E_0 \rightarrow_{L_1} \ldots E_{n-1} \rightarrow_{L_n} E_n$ 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 $t' \in E_0$ 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 $t \in \mathcal{S}(E_0, t')$.*

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

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

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

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

Proposition 24. *For every finite normalized set E of messages and normalized message g, $g \in \mathit{forge}(E)$ implies that there exists a well-formed derivation from E with goal g.*

Proof. Let $E_0 = E$ and $D = E_0 \rightarrow_{L_1} \cdots \rightarrow_{L_n} E_n$ be a derivation of goal g of minimal length. We prove that D satisfies (1) and (2) in Lemma 23. We first show:

Claim. If $F \to_{L_o(t)} F, t \to_{L_o(u)} F, t, u$, then $F \to_{L_o(u)} F, u$.

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, $F \to_{L_o(u)} F, u$. 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 $L_i = F \to t$, 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 23:

(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 minimality of *D* (otherwise L_j could be removed). 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 $j < i$ such that L_j is a t'-rule. If $L_j \in \mathcal{L}_d(t')$, again, we are done. Otherwise, $L_j \in L_{oc}(t')$, a contradiction to the choice of t'.

(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 $t \in \mathcal{S}(t')$, and if $L_j \in L_o(t')$, then since *t* is standard, we have that *t* is a factor of t', and thus, $t \in \mathcal{S}(t')$, or there exists $t'' \in E_{j-1}$ non-standard with $t \in \mathcal{S}(t'')$ (*t* is used to simplify t'') such that, 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 or L_d. Thus, $t'' \in E_0$.
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 $t \in \mathcal{S}(t')$. □

Proposition 25. *The set L_o of XOR rules is a set of oracle rules.*

Proof. We check each condition in Definition 7:

(1) The first point is a consequence of Proposition 24.
(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 $\varepsilon(u) = 0$. 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, $\varepsilon(u) \in \mathit{forge}(F)$. Let $\delta = [u \leftarrow 0]$. There are three cases:
   (a) Either $u = t$. Then, $t\delta = 0 \in \mathit{forge}(F\delta)$.
   (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 $u = \mathrm{XOR}(t_1, \dots, t_n)$ with $t_1, \dots, t_n \in F \setminus u$. 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 $\ulcorner F\delta \urcorner \to_{L_o(t\delta)} \ulcorner F\delta, t\delta \urcorner$. □

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

Proposition 26. *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, with set E, t represented as a DAG $\mathcal{G}$, 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, both can be represented as subsets of nodes of $\mathcal{G}$. Obviously, 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 $\mathit{Factor}(t) \subseteq B$. 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 $\mathit{Factor}(t)$ 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. □

As an immediate consequence of Theorem 8 we obtain that INSECUREwith XOR rules is in NP. NP-hardness can be obtained as in [27]. Altogether this yields:

Theorem 27. INSECURE *w.r.t. the XOR intruder is an NP-complete problem.*

Together with Proposition 26, Theorem 9 implies:

Theorem 28. *For the XOR intruder, the problem* DERIVE *is in PTIME.*

In [12], this problem is called *ground reachability problem* and is only 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 8, again we can show that INSECURE is NP-complete. Section 7.2.1 provides an example that illustrates the intruder's additional power.

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 8 also holds in this simplified setting.

7.2.1. Motivation

As an example, we use a variant of the Needham–Schroeder symmetric key authentication protocol [24], which is given as follows:

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

This protocol is considered to be secure in [10]. However, a careful analysis of this protocol reveals a flaw in case encryption is carried out by cipher-block-chaining (CBC) and all atoms are of the size of a block [26]. The attack in [26] exploits that if a message *m* is encrypted using the CBC mode, then it is easy to obtain an encryption of the prefix of *m* under the same key even without knowledge of the encryption key. In other words, the intruder can perform the following intruder rule:

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

In the example above, the intruder can forge $\{N_A, B\}^s_{K_{AS}}$ by applying such a rule on the second message of the protocol, 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:

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

7.2.2. Prefix rules are oracle rules

Definition 29. 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 $K, M, M_1, \ldots, M_n$, ($n \geqslant 1$). 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 8.

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

Proposition 30. *For all $t \in forge(E)$, there exists a well-formed derivation from E with goal t.*

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

(1) If $i < j$ with $L_j \in L_{oc}(\{M\}^s_K)$ and $L_i \in L_c(\{\langle ..\langle M, M'_1\rangle \ldots, M'_p\rangle\}^s_K)$, 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 $i < j$ with $L_i = \{M''\}^s_K \rightarrow \{M'\}^s_K \in L_{oc}$ and $L_j = \{M'\}^s_K \rightarrow \{M\}^s_K \in L_{oc}$, then replace L_j by the rule $\{M''\}^s_K \rightarrow \{M\}^s_K$. 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. (This size is bounded by the biggest term in the derivation.)
(3) If $i < j$ with $L_i = \{M'\}^s_K \rightarrow \{M\}^s_K \in L_{oc}$ 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 removes rules that produce messages not used in the derivation.)

Clearly, this transformation system terminates: This can easily be shown by defining a (well-founded) lexicographical ordering with the different components defined according to the remarks provided along with the transformations. Then, it is easy to observe that with every application of a transformation 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 transformation rules and eliminating redundant rules is in fact a derivation from E with goal g. We show that $D' = E'_0 \rightarrow_{L'_1} \cdots \rightarrow_{L'_m} E'_m$ 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 transformation rule (3). Therefore, we have $s \in E$ or there exists $L'_j \in \mathcal{L}_d(s)$ with $j < i$ in D'. Iterating this argument, it follows that s is a subterm of E.

For $\mathcal{L}_c$ rules, we will reason by induction on $m - i$, i.e., the induction hypothesis is that for any $m \leqslant j > i$ and any message t' with $L'_j \in \mathcal{L}_c(t')$ the condition on composition rules in well-formed derivation is satisfied. Now, assume that $L'_i \in \mathcal{L}_c(t)$. If $m - i = 0$, then $t = g$, and therefore, $t \in \mathcal{S}(E, g)$. For the induction step, there exists a rule L'_j, $j > i$, in D' using t, by the transformation 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 transformation rule (3). Thus, in any case, $L'_j \notin L_d(t)$. Using transformation rules (1) and (2), we can also conclude that $L'_j \notin L_{oc}$: While for the case that $L'_i \in L_c(t)$, this follows immediately by transformation (1), the case $L'_i \in L_{oc}(t)$ is covered by transformation (2). Consequently, $L'_j \in L_c(t')$, and t is a subterm of t'. By the induction hypothesis and since $j > i$, we know that $t' \in \mathcal{S}(g, E)$, and thus, $t \in \mathcal{S}(g, E)$. □

We can now prove that these rules are oracle rules:

Proposition 31. *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 Proposition 30.

(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 transformation rule (3) in the proof of Proposition 30 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 $\varepsilon(\langle a, b\rangle) = a$ and $\varepsilon(u) = 0$ 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 $\theta = [u \leftarrow \varepsilon(u)]$. 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 $M, K \in F\backslash u$, 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 $\{..\langle M, M'_1\rangle \ldots, M'_p\}^s_K$, $1 \leqslant q \leqslant p$ and $\langle ..\langle M, M'_1\rangle \ldots, M'_q\rangle = u$, 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 $\{M\}^s_K\theta \in forge(\{t\}^s_K\theta)$. □

Obviously, $E \rightarrow t \in L_o$ can be decided in polynomial time in $|E, t|$. Also, analogously to the proof in [27] one can show that INSECUREis NP-hard. Now, by Theorem 8, it follows:

Theorem 32. INSECURE *w.r.t. the prefix intruder is an NP-complete problem.*

With Theorem 9 we obtain:

Theorem 33. *For the prefix intruder, the problem* DERIVE *is in PTIME.*

8. Conclusion

Based on a general framework in which we equip the intruder with oracle rules, we have shown that when extending the standard Dolev–Yao intruder by (i) rules for XORing messages or (ii) rules which allow the intruder to make use of prefix properties of encryption algorithms 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 other kinds of intruder rules and algebraic laws such as those for RSA encryption and Diffie–Hellman exponentiation. First results have appeared in [7,8] is an important step in this direction. We have shown that security is decidable for a large class of protocols based on Diffie–Hellman key construction techniques, by reducing the problem to solving linear diophantine equations.

References

[1] R.M. Amadio, W. Charatonik, On name generation and set-based analysis in the Dolev–Yao Model, in: L. Brim, P. Jancar, M. Kretinsky, A. Kucera (Eds.), 13th Internat. Conf. on Concurrency Theory (CONCUR 2002), Lecture Notes in Computer Science, Vol. 2421, Springer, Berlin, 2002, pp. 499–514.

[2] R. Amadio, D. Lugiez, On the reachability problem in cryptographic protocols, in: C. Palamidessi (Ed.), Proc. CONCUR 2000—Concurrency Theory, 11th Internat. Conf., Lecture Notes in Computer Science, Vol. 1877, Springer, Berlin, 2000, pp. 380–394.

[3] A. Armando, L. Compagna, P. Ganty, SAT-based model-checking of security protocols using planning graph analysis, in: Proc. 12th Internat. FME Symp. (FME 2003), Lecture Notes in Computer Science, Vol. 2885, Springer, Berlin, 2003, pp. 875–893.

[4] D. Basin, S. Mödersheim, L. Viganò, An on-the-fly model-checker for security protocol analysis, in: E. Snekkenes, D. Gollmann (Eds.), Proc. Eighth European Symp. on Research in Computer Security (ESORICS 2003), Lecture Notes in Computer Science, Vol. 2808, Springer, Berlin, 2003, pp. 253–270.

[5] M. Boreale, Symbolic trace analysis of cryptographic protocols, in: Automata, Languages and Programming, 28th Internat. Colloq. (ICALP 2001), Lecture Notes in Computer Science, Vol. 2076, Springer, Berlin, 2001, pp. 667–681.

[6] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, An NP decision procedure for protocol insecurity with XOR, in: Proc. 18th Ann. IEEE Symp. on Logic in Computer Science, Lecture Notes in Computer Science 2003, IEEE, Computer Society Press, 2003, pp. 261–270.

[7] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, Deciding the security of protocols with Diffie–Hellman exponentiation and products in exponents, in: P.K. Pandya, J. Radhakrishnan (Eds.), FSTTCS 2003: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science, Vol. 2914, Springer, Berlin, 2003, pp. 124–135.

[8] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, Deciding the security of protocols with commuting public key encryption, in: IJCAR 2004 Workshop W6 ARSPA Automated Reasoning for Security Protocol Analysis, 2004, ENTCS.

[9] Y. Chevalier, L. Vigneron, Automated unbounded verification of security protocols, in: Proc. 14th Internat. Conf. on Computer Aided Verification (CAV 2002), Lecture Notes in Computer Science, Vol. 2404, Springer, Berlin, 2002, pp. 324–337.

[10] J. Clark, J. Jacob, A Survey of Authentication Protocol Literature, 1997. Web Draft Version 1.0 available from http://citeseer.nj.nec.com/.

[11] H. Comon, V. Cortier, J. Mitchell, Tree automata with one memory, set constraints, and ping-pong protocols, in: Automata, Languages and Programming, 28th Internat. Colloq. (ICALP 2001), Lecture Notes in Computer Science, Vol. 2076, 2001, pp. 682–693.

[12] H. Comon-Lundh, V. Shmatikov, Intruder deductions, constraint solving and insecurity decision in presence of exclusive or, in: Proc. 18th Ann. IEEE Symp. on Logic in Computer Science, Lecture Notes in Computer Science 2003, IEEE, Computer Society Press, 2003, pp. 271–280.

[13] D. Dolev, A.C. Yao, On the security of public-key protocols, IEEE Trans. Inform. Theory 29 (2) (1983) 198–208.

[14] N.A. Durgin, P.D. Lincoln, J.C. Mitchell, A. Scedrov, Undecidability of bounded security protocols, in: Workshop on Formal Methods and Security Protocols (FMSP'99), 1999.

[15] M.P. Fiore, M. Abadi, Computing symbolic models for verifying cryptographic protocols, in: 14th Computer Security Foundations Workshop (CSFW-14), IEEE, Computer Society Press, 2001, pp. 160–173.

[16] D. Kapur, P. Narendran, L. Wang, An E-unification Algorithm for Analysing Protocols that Use Modular Exponentiation, in: R. Nieuwenhuis (Ed.), Proc. 14th Internat. Conf. on Rewriting Techniques and Applications (RTA 2003), Lecture Notes in Computer Science, Vol. 2706, Springer, 2003, pp. 165–179.
[17] R. Küsters, On the decidability of cryptographic protocols with open-ended data structures, in: L. Brim, P. Jancar, M. Kretinsky, A. Kucera (Eds.), 13th Internat. Conf. on Concurrency Theory (CONCUR 2002), Lecture Notes in Computer Science, Vol. 2421, Springer, Berlin, 2002, pp. 515–530.
[18] G. Lowe, An attack on the Needham–Schroeder public-key authentication protocol, Inform. Process. Lett. 56 (1995) 131–133.
[19] G. Lowe, Breaking and fixing the Needham–Schroeder public-key protocol using FDR, in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS 1996), Lecture Notes in Computer Science, Vol. 1055, Springer, Berlin, 1996, pp. 147–166.
[20] G. Lowe, Towards a completeness result for model checking of security protocols, J. Comput. Security 7 (2–3) (1999) 89–146.
[21] C. Meadows, P. Narendran, A unification algorithm for the group Diffie–Hellman protocol, in: Workshop on Issues in the Theory of Security (WITS 2002), 2002.
[22] J.K. Millen, V. Shmatikov, Constraint solving for bounded-process cryptographic protocol analysis, in: Proc. Eighth ACM Conf. on Computer and Communications Security, ACM Press, New York, 2001, pp. 166–175.
[23] J.C. Mitchell, V. Shmatikov, U. Stern, Finite-state analysis of ssl 3.0, in: Seventh USENIX Security Symposium, 1998, pp. 201–216.
[24] R. Needham, M. Schroeder, Using encryption for authentication in large networks of computers, Commun. ACM 21 (12) (1978) 993–999.
[25] L.C. Paulson, Mechanized proofs for a recursive authentication protocol, in: 10th IEEE Computer Security Foundations Workshop (CSFW-10), IEEE Computer Society Press, 1997, pp. 84–95.
[26] O. Pereira, J.-J. Quisquater, On the perfect encryption assumption, in: Workshop on Issues in the Theory of Security (WITS 2000), 2000, pp. 42–45.
[27] M. Rusinowitch, M. Turuani, Protocol insecurity with finite number of sessions is NP-complete, in: 14th IEEE Computer Security Foundations Workshop (CSFW-14), IEEE Computer Society, 2001, pp. 174–190.
[28] P.Y.A. Ryan, S.A. Schneider, An attack on a recursive authentication protocol, Inform. Process. Lett. 65 (1) (1998) 7–10.
[29] D.X. Song, S. Berezin, A. Perrig, Athena: a novel approach to efficient automatic security protocol analysis, J. Comput. Security 9 (1/2) (2001) 47–74.
[30] S.D. Stoller, A bound on attacks on authentication protocols, in: R. Baeza-Yates, U. Montanari, N. Santoro (Eds.), Proc. Second IFIP Internat. Conf. on Theoretical Computer Science: Foundations of Information Technology in the Era of Network and Mobile Computing, Kluwer, 2002, pp. 588–600.