AVISPA

www.avispa-project.org

IST-2001-39252

Automated Validation of Internet Security Protocols and Applications

Deliverable D2.2: Algebraic Properties

Abstract

The current automated methods for protocols analysis assume the so-called *perfect encryption hypothesis*. In other words they assume that there are no relations between the messages apart from the standard ones entailed by the Dolev-Yao model. A first step towards a less abstract model is to take into account some algebraic properties of the cryptographic primitives. We show in this deliverable how to extend our analysis techniques in order to account for important algebraic properties satisfied by protocol primitives.

Deliverable details

Deliverable version: *v1.0*
Person-months required: *15*
Date of delivery: *30.07.2004*
Due on: *30.06.2004*
Classification: *public*
Total pages: *26*

Project details

Start date: *January 1st, 2003*
Duration: *30 months*
Project Coordinator: *Alessandro Armando*
Partners: *Università di Genova, INRIA Lorraine, ETH Zürich, Siemens AG*

Information Society Technologies

Project funded by the European Community under the *Information Society Technologies* Programme (1998-2002)

1 Introduction

While most methods for the formal analysis of security protocols abstract from low-level properties, i.e., certain algebraic properties of encryption, such as the multiplicativity of RSA or the properties induced by chaining methods for block ciphers, many real attacks and protocol weaknesses rely on these properties. A typical example was provided by Ryan and Schneider [26] where they give a simple attack on Bull's recursive authentication protocol: the protocol is used to distribute a connected chain of keys linking all the nodes from the originator to the server, but if one key is compromised the others can be compromised too thanks to the properties of XOR. Conversely, if XOR is considered as a free operator then, as shown by L. Paulson using the Isabelle prover [22], 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 . See e.g. [2, 25, 19]. All these results assume encryption to be perfect (*perfect encryption assumption*): One needs a decryption key to extract the plain text from the cipher text, and also, a cipher text can be generated only with the appropriate key and message (no collision). Only very few works on formal analysis have relaxed this assumption. For instance in [18], unification algorithms are designed for handling properties of Diffie-Hellman cryptographic systems. Some procedures for detecting attacks in such systems are also given in [3, 20].

In this deliverable, we report on some generalisations of the decidability result of [25], stating that insecurity for finitely many protocol sessions is in NP, to some cases where messages may contain operators with algebraic properties and where the Dolev-Yao intruder is extended by the ability to compose and decompose messages with these operators. More precisely, we give a linear bound on the size of messages exchanged in minimal attacks and present an NP procedure for deciding insecurity. The main case that we consider (and which is given with details) is the one of the XOR operator. The decidability result for XOR 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 these results, 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. We show also that the Dolev-Yao intruder equipped with the ability to exploit prefix properties of encryption algorithms based on cipher-block-chaining (CBC) falls into our framework as well. An attack on the Needham-Schroeder symmetric key authentication protocol [21] exploiting this prefix property is described by Pereira and Quisquater in [23].

The decidability results presented here (see also [7]) are the first, besides the ones by Comon and Shmatikov [11], that go beyond the perfect encryption assumption. We briefly compare our work with [11]: As an immediate consequence of our proof, the problem of checking whether a message can be derived by an intruder in presence of the XOR operator — this problem is called *ground reachability* in [11] — is in PTIME. In [11], this problem

is shown to be in NP both for the case of XOR and the case of Abelian Groups. As for the general insecurity problem, we show NP-completeness based on a theorem that ensures the existence of attacks of linear size. Comon and Shmatikov present a decision procedure with a higher complexity. This procedure is based on constraint solving techniques. However, they consider a more general class of protocol rules.

It is also possible with the same technique to simulate an intruder that can exploit exponentiation function properties and therefore we are able to find attacks on protocols based on Diffie-Hellman key exchange technique. Finally, we can take into account the commutation properties of public keys for some protocols based on RSA when the same modulus is employed for defining the keys.

In the following Section 2, we provide examples illustrating the role of algebraic properties of operators such as XOR in attacks. We then introduce our protocol and intruder model in Section 3. In particular, we introduce a notion of oracle rules to extend the capabilities of intruders. The decidability result for the general framework is presented (for detailed proofs see [8]), including the description of the NP decision algorithm. As instances of this framework we present XOR rules and prefix rules and the related decidability and complexity results. We briefly discuss an extension of the result to modular exponentiation in Section 4 and conclude in Section 5.

2 Examples of Algebraic Properties in Protocols

In this section, we present a few typical examples of protocols and/or attacks based on using algebraic protocols of cryptographic primitives. They are given in a common and intuitive syntax, namely as a sequence of messages with a specification of receivers and senders. In general we denote by {M}K the encryption of message M with key K, and {M,M'}K the encryption using key K of the compound message built by concatenating M and M'.

2.1 Commutative encryption

Shamir Three Pass Protocol

AUTHOR(S): A. Shamir (see [27], Subsection 22.3.) 1992

SUMMARY: The following protocol [10] enables to exchange a secret message without sharing any initial secret.

PROTOCOL SPECIFICATION

```
A, B :    principal
Ka, Kb :  key
   1.    A  ->  B  :    {M}Ka
   2.    B  ->  A  :    {{M}Ka}Kb
   3.    A  ->  B  :    {M}Kb
```

REQUIREMENTS: This protocol assumes that encryption is commutative, i.e. {{x}y}z = {{x}z}y.

DESCRIPTION OF THE PROTOCOL RULES: The agent A encrypts his message M by Ka, then B encrypts the message he received by Kb. Since

$$\{\{M\}Ka\}Kb=\{\{M\}Kb\}Ka,$$

the agent A can decrypt it and send {M}Kb to B. Then, using his key Kb, B can retrieve M.

ATTACKS: This protocol is subject to a variety of attacks [10]. The Man-in-The-Middle attack that we present here exploits the absence of authentication mechanisms between participants.

```
1.     A    ->  B   :    {M}Ka
2.   I(B)  ->  A   :    {{M}Ka}Ki
3.     A    ->  B   :    {M}Ki
```

Now, the intruder can compute the message M.

2.2 XOR

The $\oplus$ symbol denotes the binary operation called XOR. The properties of XOR are:

- $x \oplus (y \oplus z) = (x \oplus y) \oplus z$ (associativity),
- $x \oplus y = y \oplus x$ (commutativity),
- there is an element 0 such that $x \oplus 0 = 0 \oplus x = x$, and
- $x \oplus x = 0$ (inverse),

This operation is used in many protocols like the John Bull protocol of APM, the standard 802.11 protocol or the TMN protocol, presented in Section 2.4.

John Bull Protocol of APM using XOR

AUTHOR(S): P. Ryan, S. Schneider 1998

SUMMARY: The protocol proposed by J. Bull in [5] establishes a chain of session keys between a fixed number of participants and a server. The following description of the protocol uses just the three principals Alice, Bob and Charlie.

PROTOCOL SPECIFICATION

```
A, B, C, S :  principal
Ka, Kb, Kc :  key
hashK{X} :      message -> hash
SignK{X} :      message -> message, hash
```

```
1.   A  ->  B  :   SignKa{A,B,Na,-} = Xa
2.   B  ->  C  :   SignKb{B,C,Nb,-} = Xb
3.   C  ->  S  :   SignKc{C,S,Nc,-}
4.   S  ->  C  :   A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab,
                   B,A,Kab ⊕ hashKb{Nb},{B,A,Nb}Kab,
                   B,C,Kbc ⊕ hashKb{Nb},{B,C,Nb}Kbc,
                   C,B,Kbc ⊕ hashKc{Nc},{C,B,Nc}Kbc
5.   C  ->  B  :   A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab,
                   B,A,Kab ⊕ hashKb{Nb},{B,A,Nb}Kab,
                   B,C,Kbc ⊕ hashKb{Nb},{B,C,Nb}Kbc
6.   B  ->  A  :   A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab
```

Note: In the declaration part, `hashK{X} message -> hash` should be understood as parametric in `K`. The same remark holds for `SignK{X}`.

DESCRIPTION OF THE PROTOCOL RULES: Each principal takes its inputs, appends a nonce and some address information and signs the result using a hash keyed with its private key. He sends the result to the next node. The last node is the server. He can verify all the received and create some new session keys which it conceals by exclusive-oring them with hashes that only the intended recipients can compute. These new messages are sent by the same way to all the principals.

ATTACKS: The attack described in [26] is called *domino attack*. The compromise of any key leads to the compromise of all other keys. The cause of this attack is the fact that the server sends back a message like:

```
Kab ⊕ hashKa{Na},Kab ⊕ hashKb{Nb}
Kbc ⊕ hashKb{Nb},Kbc ⊕ hashKc{Nc}
```

Now, anyone can compute:

```
Kab ⊕ hashKb{Nb} ⊕ Kbc ⊕ hashKb{Nb} = Kab ⊕ Kbc
```

So if only one principal knows a key, he knows all the keys "downstream" from it. This is why this attack is called the *domino attack*.

REMARK: This protocol assumes that all the principals are honest.

Standard 802.11 in WEP (Wired Equivalent Privacy) protocol

AUTHOR(S): N. Borisov, I. Goldberg, D. Wagner 2001

SUMMARY: The Wired Equivalent Privacy protocol is used in 802.11 networks to protect link-level data during wireless transmission. It is described in detail in the 802.11 standard [15].

PROTOCOL SPECIFICATION

```
A, B :  principal
K :     key
RC4 :   initial vector, key -> message
C :     message -> integrity
   1.    A  ->  B   :     v, ( [M,C(M)] ⊕ RC4(v,K) )
```

DESCRIPTION OF THE PROTOCOL RULES: The two principals share a key K. A wants to communicate the message M to B. The agent A takes an initial vector v and computes RC4(v,K). The RC4 algorithm generates a *keystream* (i.e. a long sequence of pseudorandom bytes) as a function of v and K. The agent A applies the XOR operator to RC4(v,K) and P = `[M,C(M)]`, where C(M) is the *integrity checksum* of the message M. Then B can decrypt the message because he knows K and has received v.

REQUIREMENTS: The checksum function C has the following homomorphic property on the $\oplus$ symbol: C(X $\oplus$ Y) = C(X) $\oplus$ C(Y).

ATTACKS: The two attacks described in [4] are based on the property of the XOR operator for the first one and on the homomorphic property of the function C for the second one.

The first attack uses the fact that if the encryption of the plain text P1 is C1=P1 $\oplus$ RC4(v,K) and the encryption of the plain text P2 is C2=P2 $\oplus$ RC4(v,K) with the same initial vector v, then C1 $\oplus$ C2 = P1 $\oplus$ P2. So if an intruder knows a plain text and his cipher he can decrypt any message using the same key K.

The second attack consists in modifying a message without the receiver noticing it. Let M' = M $\oplus$ D and C an intercepted cipher. The attacker computes:

$$\begin{aligned} C \oplus \texttt{[D,C(D)]} &= RC4(v,K) \oplus \texttt{[M,C(M)]} \oplus \texttt{[D,C(D)]} \\ &= RC4(v,K) \oplus \texttt{[M} \oplus \texttt{D,C(M)} \oplus \texttt{C(D)]} \\ &= RC4(v,K) \oplus \texttt{[M} \oplus \texttt{D,C(M} \oplus \texttt{D)]} \\ &= RC4(v,K) \oplus \texttt{[M',C(M')]} \\ &= C' \end{aligned}$$

It is possible to add any pair composed by D and C(D) to a cipher without the receiver noticing this change.

REMARK: The function checksum is implemented by the CRC-32, and this function has the required homomorphic property mentioned above.

2.3 Abelian Group

Let the + symbol denote the additive binary operation of the Abelian Group (G,+). The properties of the Abelian Group are:

- For all x, y, z $\in$ G, x + (y + z) = (x + y) + z (associativity).
- There is an element 0 $\in$ G such that x + 0 = 0 + x for all x $\in$ G.

- For all x, $\in$ G there exists y such that x + y = 0 (inverse)
- For all x, y $\in$ G, x + y = y + x (commutativity)

We present a protocol for computing an average, using these properties. Lots of protocols use the structure of the Abelian Groups in combination with other properties.

Average Salary Computation

AUTHOR(S): B. Schneier 1996

SUMMARY: A group of people wants to compute the average of their salaries without anyone declaring his own salary to the others. It is necessary to suppose that each principal has a public key. This protocol is described by [27]. The protocol is given by example for principals Alice, Bob, Charlie and Dorothy.

PROTOCOL SPECIFICATION

```
A, B, C, D :      principal
Ka, Kb, Kc, Kd :  key
PK :              principal -> key
   1.    A  ->    B     :     A, {Na + Sa}PK(B)
   2.    B  ->    C     :     B, {Na + Sa + Sb}PK(C)
   3.    C  ->    D     :     C, {Na + Sa + Sb + Sc}PK(D)
   4.    D  ->    A     :     D, {Na + Sa + Sb + Sc + Sd}PK(A)
   5.    A  ->  B,C,D   :     ( Sa + Sb + Sc + Sd ) / 4
```

ATTACKS: This protocol assumes that every principal is honest. If the first principal is malicious he can lie about his salary and so the average will be wrong.

2.4 Homomorphism

In this section, we consider operators that verify equalities of the form $f(g(x, y)) = g(f(x), f(y))$.

Needham-Schroeder-Lowe Protocol with ECB

AUTHOR(S): R. Needham, M. Schroeder, G. Lowe 1995

SUMMARY:

- Electronic CodeBook (ECB) mode is the most obvious way to use a block cipher. A block cipher encrypts plain text in fixed-sized-n-bits blocks (often n=64). For messages exceeding n bits, the simplest approach is ECB. ECB consists in partitioning the message into n-bits blocks and encrypting each of them separately.

- The well-known Needham-Schroeder Public-Key Authentication Protocol corrected by G. Lowe [16] who proved that it is correct with perfect encryption.

PROTOCOL SPECIFICATION

```
A, B :    principal
Na, Nb :  nonce
PK :      principal -> key
   1.    A  ->  B  :    {A, Na}PK(B)
   2.    B  ->  A  :    {B, Na, Nb}PK(A)
   3.    A  ->  B  :    {Nb}PK(B)
```

DESCRIPTION OF THE PROTOCOL RULES: A sends to B his name and a new nonce encrypted by the public key of B. B confirms to A with the message encrypted by A's public key composed with B's identity, the nonce received and his new nonce. A sends back the nonce of B encrypted by B's public key. Each principal is sure to talk with the right principal. They share the secrets Na and Nb.

REQUIREMENTS: A basic property of ECB is that

$$\{ABC\}k = \{A\}k\{B\}k\{C\}k$$

where the size of the blocks A, B, C is the maximal size accepted by the cryptographic algorithm.

ATTACKS: An intruder can attack the protocol by playing two sessions of the protocol and extracting {Na, Nb}PK(A) from {B, Na, Nb}PK(A). He reuses it to compute {I, Na, Nb}PK(A) and trick A to decrypt Nb for him.

```
   i.1.      A    ->    I    :    {A, Na}PK(I)
   ii.1.   I(A)   ->    B    :    {A, Na}PK(B)
   ii.2.     B    ->  I(A)   :    {B, Na, Nb}PK(A)
                             The intruder extracts {Na, Nb}PK(A)
   i.2.      I    ->    A    :    {I, Na, Nb}PK(A)
   i.3.      A    ->    I    :    {Nb}PK(I)
   ii.3.     I    ->    B    :    {Nb}PK(B)
```

REMARK: This attack is possible only if the ECB mode is used, and if the size of every nonce and principal name is a multiple of the maximal size of the cipher imposed by the encryption function.

2.5 Prefix property

The prefix property is the ability for an intruder to get any prefix of an encrypted message: from a message {x,y}z, he can deduce the message {x}z. This property strongly depends on the encryption algorithm. For example, the ECB algorithm, presented in section 2.4

has the property. But the ECB algorithm is not widely used. A relatively good method of encrypting several blocks of data is Cipher Block Chaining (CBC). In such system, the encryption of a message block sequence $P_1P_2\cdots P_n$ with the key K is $C_0C_1C_2\cdots C_n$ where $C_0 = I$ (initialisation block) and $C_i = \{C_{i-1} \oplus P_i\}_K$. The CBC encryption system has the following property: if $C_0C_1C_2\cdots C_iC_{i+1}\cdots C_n = \{P_1P_2\cdots P_iP_{i+1}\cdots P_n\}_K$ then $C_0C_1C_2\cdots C_i = \{P_1P_2\cdots P_i\}_K$, which we call the prefix property.

We show how this property can be exploited to build an attack against the Denning-Sacco Symmetric-Key Protocol and the Needham-Schroeder Symmetric-Key Protocol.

Denning-Sacco Symmetric-Key Protocol with CBC

AUTHOR(S): D. E. Denning, G. M. Sacco 1981

SUMMARY: Modified version [12] of the Needham-Schroeder Symmetric-Key Protocol with timestamps to fix the freshness flaw. Distribution of a shared symmetric key by a trusted server and mutual authentication.

PROTOCOL SPECIFICATION

```
A, B, S :          principal
Kas, Kbs, Kab :    key
T :                timestamp
   1.    A  ->  S   :    A, B
   2.    S  ->  A   :    {B, Kab, T, {A, Kab, T}Kbs}Kas
   3.    A  ->  B   :    {A, Kab, T}Kbs
```

DESCRIPTION OF THE PROTOCOL RULES: The shared symmetric key established by the protocol is `Kab`.

ATTACKS: Y. Chevalier and L. Vigneron [9] present an attack on this protocol. It is based on the fact that messages 2 and 3 follow the same pattern. We also assume that all components of the messages have length one block. This vulnerability is unlikely to occur in practice but it is nevertheless important to be sure that actual implementations do not suffer from it.

```
   1.    I(B)  ->   S    :    B, A
   2.     S    ->  I(B)  :    {A, Kab, T, {B, Kab, T}Kas}Kbs
   3.    I(A)  ->   B    :    {A, Kab, T}Kbs
```

Using the answer `{A, Kab, T, {B, Kab, T}Kas}Kbs` of the server and the prefix property, the intruder is able to get the message `{A, Kab, T}Kbs`. Receiving this message, B accepts a new value for the symmetric key he shares with A, whereas A is not aware that a protocol run took place.

Needham-Schroeder Symmetric Key Authentication Protocol with CBC

AUTHOR(S): R. Needham, M. Schroeder 1978

SUMMARY: Distribution of a shared symmetric key by a trusted server and mutual authentication [21].

PROTOCOL SPECIFICATION

```
A, B, S :        principal
Na, Nb :         nonce
Kas, Kbs, Kab :  key
dec :            nonce -> nonce
   1.    A  ->  S   :    A, B, Na
   2.    S  ->  A   :    {Na, B, Kab, {Kab, A}Kbs}Kas
   3.    A  ->  B   :    {Kab,A}Kbs
   4.    B  ->  A   :    {Nb}Kab
   5.    A  ->  B   :    {dec(Nb)}Kab
```

DESCRIPTION OF THE PROTOCOL RULES: This protocol establishes the fresh shared symmetric key Kab. Messages 1-3 perform the distribution of the fresh shared symmetric key Kab and messages 4-5 are for mutual authentication of A and B. The operator dec is decrementation.

REQUIREMENTS: The protocol must guarantee the secrecy of Kab: in every session, the value of Kab must be known only by the participants playing the roles of A, B and S in that session. If the participant playing B accepts the last message 5, then Kab has been sent in message 3. by A (whose identity is included in the cipher of message 3).

ATTACKS: Beyond other existing attacks, O. Pereira and J.-J. Quisquater [23] presented the following flaw, based on the prefix property.

```
i.1.      A    ->    S     :    A, B, Na
i.2.      S    ->    A     :    {Na, B, Kab, {Kab, A}Kbs }Kas
ii.3.   I(B)   ->    A     :    {Na, B}Kas
ii.4.     A    ->  I(B)    :    {Naa}Na
ii.5.   I(B)   ->    A     :    {dec(Naa)}Na
```

Suppose that the message {Na, B, Kab, {Kab, A}Kbs}Kas has cipher text $C_0C_1C_2\cdots C_n$ and that all components have length one block. Then {Na, B}Kas has cipher text $C_0C_1C_2$. This message is of the form of an expected message by A at the third step of a session initialised by B. Thus A can be fooled into accepting the publicly known nonce Na as a secret key shared with B.

3 Analysing protocols in presence of XOR

3.1 The Protocol and Intruder Model

The protocol and intruder model we describe here extends standard models for the (automatic) analysis of security protocols [1, 14, 25, 19] 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 mentioned oracle rules. In what follows, we provide a formal definition of our model by defining terms, messages, protocols, the intruder, and attacks.

3.1.1 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 (respectively private) key k to its corresponding private (respectively 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 $E, F, ...$, 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 $\mathcal{F}(t) = \uplus_{t' \in M} \mathcal{F}(t')$, 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 $\text{XOR}(a, b)$ is not a subterm of $\text{XOR}(\text{XOR}(a, b), c)$.

We define the size of a term and of a set of terms basically as the size of their representation as a *Directed Acyclic Graph (DAG)*. That is, the *(DAG) size* $|t|$ (respectively $|E|$) of a term t (respectively a set of terms E) is the cardinality of the set $\mathcal{S}(t)$ (respectively $\mathcal{S}(E)$). 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 such that 0 is the unit element. According to these properties, the normal form of a term is defined as the result of the *normalisation 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 normalisation function is recursively defined as follows:

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

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

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

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

One easily shows:

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

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

We finally remark:

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

3.1.2 Protocols

The following definition is explained below.

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

A protocol *P is a tuple $(\{R_i \Rightarrow S_i,\ i \in \mathcal{I}\}, <_{\mathcal{I}}, E)$ where E is a finite normalised 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 normalised;*
2. *for all $x \in \mathcal{V}(S_i)$, there exists $j \leq_{\mathcal{I}} i$ such that $x \in \mathcal{V}(R_j)$;*
3. *for every subterm* 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 normalised, $t_1, \ldots, t_n$ are standard terms.)*

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

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

Intuitively, when executing a rule $R_i \Rightarrow S_i$ and on receiving a (normalised) message m in a protocol run, it is first checked whether m and R_i match, i.e., whether there exists a ground substitution σ such that $m =_{\text{XOR}} R_i\sigma$. If so, $\ulcorner S_i\sigma \urcorner$ is returned as output. We always assume that the messages exchanged between principals (and the intruder) are normalised — therefore, m is assumed to be normalised 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 normalised terms (representing the corresponding equivalence class of terms). Finally, we note that since the different protocol rules may share variables, some of the variables in R_i and S_i may be bounded already by

substitutions obtained from applications of previous protocol rules. We are not actually interested in a normal execution of a protocol but rather in attacks on a protocol. This is the reason why the definition of a protocol contains the initial intruder knowledge. Attacks are formally defined in Subsection 3.2.1.

Condition 1. in the above definition is not a real restriction since, due to Lemma 1, the transformation performed by a protocol rule and its normalised variant coincide. Condition 2. guarantees that when an output is produced with S_i, all variables in S_i are "bounded" already. Otherwise, the output of a protocol rule would be arbitrary, since unbounded variables could be mapped to any message. Condition 3. guarantees that the bounding of variables is deterministic. For example, if the protocol rule $\text{XOR}(x, y) \Rightarrow \langle x, y\rangle$ does not have predecessors according to $<_{\mathcal{I}}$, and thus x and y are not bounded, then this rule violates Condition 3: On receiving $\text{XOR}(a, b, c)$, for instance, different substitutions are possible, including $\{x \mapsto \text{XOR}(a, b), y \mapsto c\}$, $\{x \mapsto \text{XOR}(b, d), y \mapsto \text{XOR}(a, c, d)\}$, etc. In other words, a principal must guess a substitution. With Condition 3. we avoid this. We point out that in [11] no restrictions on protocol rules are put, and thus also these rather unrealistic rules are allowed.

3.2 Example

As an example, we use a variant of the Needham-Schroeder-Lowe Protocol [17], i.e., the Needham-Schroeder Public-Key Protocol with Lowe's fix, where XOR is used in some places instead of concatenation. 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:

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

If XOR is interpreted as a free symbol, such as pairing, then according to [17] this protocol is secure. In particular, the intruder is not able to get hold of N_B. However, if the algebraic properties of XOR are taken into account, the following attack is possible, which is a variant of the original attack on the Needham-Schroeder Protocol and which allows the intruder I to obtain N_B. In this attack, two sessions run interleaved where the steps of the second session are marked with $'$. In the first session, A talks to the intruder I, and in the second session I, purporting to be A, talks to B. We emphasise that in this attack I generates new messages by applying the XOR operator and uses that $\text{XOR}(N_A, B, I, B) =_{\text{XOR}} \text{XOR}(N_A, I)$.

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

The above protocol can formally be stated as follows: The set of atoms is $\mathcal{A} = \{na, a, I, b, ka, kb, ki, ki^{-1}, 0, \mathsf{secret}\}$ where the secret was N_B. The initial intruder knowledge is $S_0 = \{0, I, ki, ki^{-1}, ka, kb\}$. The protocol rules are (we have only retained the protocol steps that are used in the attack)

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

We have $\mathcal{I} = \{(a,1),(a,2),(b,1)\}$ and $<_{\mathcal{I}} := \{((a,1),(a,2))\}$.

3.2.1 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 corresponding keys. 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 Subsection 3.2.2, we consider classes of guess rules with certain properties, so-called oracle rules. As mentioned, in Subsection 3.4 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 messages 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 deliverable is depicted in Table 1. In this table, a, b denote (arbitrary) messages, K is an element of $\mathcal{K}$, and E is a finite set of messages (considered as multiset).

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

The intruder rules are denoted as shown in Table 1. With $L_{od}(a)$ and $L_{oc}(a)$ we denote (finite or infinite) sets of guess rules. For uniformity, we therefore consider

$$L_{p1}(\langle a, b\rangle), \ldots, L_{sd}(\{a\}^s_b)$$

and $L_c(\langle a, b\rangle), \ldots, L_c(\{a\}^s_b)$ as singletons. Note that, even if there are no guess rules, the number of decomposition and composition rules is always infinite since there are infinitely many messages a, b.

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

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

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

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

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

Lemma 2. *If E is a normalised set of messages, then $\mathit{forge}(E)$ is normalised.*

The lemma says that if an intruder only sees normalised messages, then he only creates normalised messages. Intruders should be modelled in such a way that they cannot distinguish between equivalent messages since if one thinks of, for instance, the message $\mathrm{XOR}(a, a, b)$, 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 normalised messages, where every single normalised 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 previous 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, [25].

Table 1: Intruder Rules

Decomposition rules		Composition rules	
$L_{p1}(\langle a,b\rangle)$:	$\langle a,b\rangle \to a$	$L_c(\langle a,b\rangle)$:	$a,b \to \langle a,b\rangle$
$L_{p2}(\langle a,b\rangle)$:	$\langle a,b\rangle \to b$		
$L_{ad}(\{a\}^p_K)$:	$\{a\}^p_K, K^{-1} \to a$	$L_c(\{a\}^p_K)$:	$a,K \to \{a\}^p_K$
$L_{sd}(\{a\}^s_b)$:	$\{a\}^s_b, b \to a$	$L_c(\{a\}^s_b)$:	$a,b \to \{a\}^s_b$
$L_{od}(a)$:	$E \to a$	$L_{oc}(a)$:	$E \to a$
with a subterm of E and E normalised.		with E, a normalised and such that every proper subterm of a is a subterm of E.	

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

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

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

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

3.2.2 Oracle Rules

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

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

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

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

We can now define oracle rules. Condition 1. in the following definition will allow us to bound the length of derivations. The remaining conditions are later used to bound the size of the substitution σ of an attack. They allow us to replace a subterm u in σ, composed by the intruder, by a smaller message.

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

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

3.3 Main Theorem and the NP Decision Algorithm

We now state the main theorem of this paper. In Subsection 3.4, this theorem will allow us to show that INSECURE is in NP in presence of an intruder that uses XOR rules and prefix rules, respectively.

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

The NP decision procedure is given in Figure 1. Clearly, the procedure is sound. To show completeness, one has to prove that if there exists an attack (π, σ) on P, then there is one with the size of σ bounded as in step 2. of the procedure. Then we can show that step 3. and 4. in the procedure can be carried out in polynomial time. More precisely, we show that the following problem, henceforth called *derivation problem*, can be solved in polynomial time in the (DAG) size of the input:

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

where E is a finite set of messages and t is a message, both given as DAGs. In the procedure, E is the set $\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner$ for some $i \in \{1, \ldots k\}$ and t is $\ulcorner R_i\sigma \urcorner$ or secret. It can be shown that $|E,t| \leq 4 \cdot |P|$, and thus, the procedure depicted in Figure 1 is in fact an NP decision procedure.

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

1. Guess an execution order π for P. Let k be the size of π.
   Let $R_i = R'_{\pi^{-1}(i)}$ and $S_i = S'_{\pi^{-1}(i)}$ for $i \in \{1, \ldots, k\}$.
2. Guess a normalised ground substitution σ such that $|\sigma(x)| \leq 4n$ for all $x \in V$.
3. Check that $\ulcorner R_i\sigma \urcorner \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < i\} \cup \{S_0\} \urcorner)$ for every $i \in \{1, \ldots, k\}$.
4. Check $\mathsf{secret} \in \mathit{forge}(\ulcorner \{S_j\sigma \mid j < k+1\} \cup \{S_0\} \urcorner)$.
5. If each check is successful, then answer "yes", and otherwise "no".

Figure 1: NP Decision Procedure for Insecurity

3.4 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.

3.4.1 XOR Rules

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

Definition 5. *We define $L_o = L_{oc} \cup L_{od}$ to be the set of* XOR rules *where*

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

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

Note that the rules in L_{od} are in fact decomposition guess rules since if $\ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ 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 normalised. Also, the rules in L_{oc} are composition guess rules since proper subterms of $\ulcorner \mathrm{XOR}(t_1, \ldots, t_n) \urcorner$ 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 normalised.

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

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

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

Example Before showing that the XOR rules are oracle rules, we illustrate that the XOR intruder can perform the attack informally described in Subsection 3.2.

We recall (see Subsection 3.1.2) that the protocol underlying the attack is formally described as follows:

The initial intruder knowledge is $S_0 = \{0, I, ki, ki^{-1}, ka, kb\}$. The protocol rules are

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

We have $\mathcal{I} = \{(a,1), (a,2), (b,1)\}$ and $<_{\mathcal{I}} := \{((a,1),(a,2))\}$.

When using a perfect encryption model, there is no attack on this instance of the protocol since the intruder is not able to forge $\{\mathsf{secret}, \mathrm{XOR}(na, I)\}^p_{ka}$ 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 $\sigma(x_{na}) = \mathrm{XOR}(na, b, I)$ and $\sigma(x_{\mathsf{secret}}) = \mathsf{secret}$ is an attack on this protocol.

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

Lemma 3. *Let $D = E_0 \to_{L_1} \ldots E_{n-1} \to_{L_n} E_n$ be a derivation with goal g such that:*

1. *For every j with $E_{j-1} \to_{L_j} E_{j-1}, t$ 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 $t \in \mathcal{S}(E, t')$.*

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

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

Proposition 2. *For every finite normalised set E of messages and normalised message g, $g \in forge(E)$ implies that there exists a well formed derivation from E with goal g.*

Proposition 3. *The set L_o of XOR rules is a set of oracle rules.*

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

Proposition 4. *Let L_o be the set of XOR rules. Then, the problem whether $E \to t \in L_o(t)$, for a given finite normalised set E of messages and a normalised 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 $Factor(t) \subseteq B$. And this set can be represented as a vector of length $|B|$ with entries 0 and 1 where an entry indicates whether a message in B belongs to $Factor(t)$ or not. This vector can be interpreted as an element of the vector space of dimension $|B|$ over the field with two elements. In the same way the terms in E can be represented. Now, deciding $E \to_{L_o(t)} E, t$ is equivalent to deciding whether the vector representing t can be represented as a linear combination of the vectors representing the messages in E. This can be done in polynomial time by Gaussian elimination. □

As an immediate consequence of Theorem 1 we obtain that INSECURE with XOR rules is in NP. NP-hardness can be obtained as in [25]. Altogether this yields:

Theorem 2. INSECURE *w.r.t. the XOR intruder is an NP-complete problem.*

From Proposition 4, this implies:

Theorem 3. *For the XOR intruder, the problem* DERIVE *is in PTIME.*

In [11], this problem is called *ground reachability problem* and is only shown to be in NP.

3.4.2 Prefix Rules

As another instance of oracle rules, we consider what we call prefix rules as in Section2. These rules allow the intruder to exploit certain properties of block encryption algorithms, based for example on cipher block chaining (CBC). Using Theorem 1, again we can show that INSECURE is an NP-complete problem.

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

Motivation As an example, we use a variant of the Needham-Schroeder Symmetric-Key Protocol [21], which is given as follows:

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

This protocol is considered to be secure in [10] whereas in [23] a careful analysis of this protocol reveals a flaw in case encryption is carried out by cipher-block-chaining and all atoms are of the size of a block. Our aim is to automate such analysis by using deduction rules of the shape:

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

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

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

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

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

Prefix rules are oracle rules.

Definition 6. *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*

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

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

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

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

Proposition 5. *For all $t \in forge(E)$, there exists a well formed derivation from E with goal t.*

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

1. If $i < j$ such that $L_j \in L_{oc}(\{M\}_K^s)$ and $L_i \in L_c(\{\langle .. \langle M, M_1' \rangle \ldots, M_p' \rangle\}_K^s)$, then replace L_j by a sequence of L_d rules decomposing $\langle .. \langle M, M_1' \rangle \ldots, M_p' \rangle$ to M followed by $L_c(\{M\}_K^s)$. Note that the number of L_{oc} rules strictly decreases.

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

3. If $i < j$ such that $L_i = \{M'\}_K^s \to \{M\}_K^s \in L_{oc}$ and $L_j \in L_d(\{M\}_K^s)$, replace L_j by $L_d(\{M'\}_K^s)$ 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 $L_{oc}(\{M'\}_K^s)$ rule in D.

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

Clearly, this transformation system terminates: This can easily be shown by defining a (well founded) lexicographical ordering with the different components defined according to the remarks added to the transformation rules above. 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 $D' = E_0' \to_{L_1'} \ldots \to_{L_m'} E_m'$ 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 $L_{i<j}' \in \mathcal{L}_d(s)$ in D'. By iteration on i, it follows that s is a subterm of E.

For $\mathcal{L}_c$ rules, we will reason by induction on $m-i$. Assume that $L_i' \in \mathcal{L}_c(t)$. If $m-i = 0$, then $t = g$, and therefore, $t \in \mathcal{S}(E, g)$. 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, $L_j' \notin L_d(t)$. By the transformation rules (1) and (2), we conclude $L_j' \notin L_{oc}$. Thus, $L_j' \in L_c(t')$, and t is a subterm of t'. By induction, $t' \in \mathcal{S}(g, E)$, and thus, $t \in \mathcal{S}(g, E)$. □

We can now prove that these rules are oracle rules:

Proposition 6. *The set L_o of prefix rules is a set of oracle rules.*

Obviously, $E \rightarrow t \in L_o$ can be decided in polynomial time in $|E, t|$. Also, analogously to the proof in [25] one can show that INSECURE is NP-hard. Now, by Theorem 1, it follows that:

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

And we also have that:

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

4 Handling (some) Protocols based on Diffie-Hellman

Using non-trivial extensions of our technique above (see also [7]), we have shown that the insecurity problem for protocols that use Diffie-Hellman exponentiation with products in exponents is NP-complete [6]. Our model is powerful enough to uncover attacks on the A-GDH.2 protocol suite discovered in [24].

Boreale and Buscemi [3] have addressed a similar problem. However, in their paper among other restrictions they put an a priori bound on the number of factors that may occur in products. In our own work we allow for an unlimited number of factors. Also, Boreale and Buscemi do not provide a complexity result. Diffie-Hellman exponentiation is also studied by Millen and Shmatikov [20]. Similar to our work, Millen and Shmatikov assume that products only occur in exponents. However, they do not provide a decision procedure. Also, they assume the base in exponentiations to be a fixed constant. This is a severe restriction since in general this rules out realistic attacks, even in case the protocol under consideration assumes a fixed basis. However recently Shmatikov [28] has been able to derive a decision procedure by reducing the problem to solving linear Diophantine equations (similarly to our approach). His result can handle a larger class of protocols than ours.

5 Conclusion

We have shown that when extending the standard Dolev-Yao intruder by rules for XOR-ing messages the protocol insecurity problem for a finite number of sessions remains NP-complete. This is the first tight complexity bound given for the insecurity problem without the perfect encryption assumption. Here we have only considered insecurity as failure of secrecy. However, we believe that our result holds also for other properties that can be reduced to reachability problems in our model, such as authentication. We need to find a way to characterise the class of intruder rules for which decidability of insecurity remains true. Finally, the decision procedures we have obtained so far are not effective. The algebraic properties are also a source of combinatorial explosion. Therefore we need a careful analysis in order to implement efficiently these decision procedures. Our idea is to rely on unification algorithms for the combination of equational theories in order to benefit from recent advances and optimizations in this domain.

References

[1] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In C. Palamidessi, editor, *Proceedings of Concur'00*, LNCS 1877, pages 380–394. Springer-Verlag, 2002.

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

[3] M. Boreale and M. Buscemi. Symbolic analysis of crypto-protocols based on modular exponentiation. In *Proceedings of MFCS '03*, LNCS 2747. Springer-Verlag, Berlin, 2001.

[4] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MOBICOM-01)*, pages 180–188, New York, 2001. ACM Press.

[5] J. Bull. The authentication protocol. *APM Report*, 1997.

[6] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science, FST TCS'03*, LNCS 2914. Springer-Verlag, 2003. Available at `http://www.avispa-project.org`.

[7] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. An NP Decision Procedure for Protocol Insecurity with XOR. In P. Kolaitis, editor, *Proceedings of LICS'2003*. IEEE, 2003.

[8] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. An NP Decision Procedure for Protocol Insecurity with XOR. Technical Report RR-4697, INRIA, 2003. URL: `http://www.inria.fr/rrrt/rr-4697.html`.

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

[10] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997. URL: `www.cs.york.ac.uk/~jac/papers/drareview.ps.gz`.

[11] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. Symp. on Logic in Computer Science (LICS'03), IEEE Computer Society Press*, 2003.

[12] D. Denning and G. Sacco. Timestamps in key distributed protocols. *Communication of the ACM*, 24(8):533–535, 1981.

[13] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

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

[15] L.M.S.C of the IEEE Computer Science Society. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, 1999.

[16] G. Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, november 1995.

[17] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *TACAS*, LNCS, pages 147–166. Springer-Verlag, 1996.

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

[19] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the ACM Conference on Computer and Communications Security CCS'01*, pages 166–175, 2001.

[20] J. K. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 47–61. IEEE Computer Society Press, 2003.

[21] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), december 1978.

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

[23] O. Pereira and J.-J. Quisquater. On the perfect encryption assumption. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS 2000), pp. 42-45. Geneva, Switzerland, July 2000*, 2000.

[24] O. Pereira and J.-J. Quisquater. Security analysis of the Cliques protocols suites. In *Proceedings of CSFW'01*. IEEE Computer Society Press, 2001.

[25] M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *Proceedings of CSFW'01*. IEEE Computer Society Press, 2001. Available at `http://www.avispa-project.org`.

[26] P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.

[27] B. Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1996.

[28] V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *13th European Symposium on Programming (ESOP '04)*, LNCS 2986. Springer-Verlag, 2004.