A Simple Constraint-solving Decision Procedure for Protocols with Exclusive or*

Yannick Chevalier

Projet CASSIS – LORIA-INRIA
email: chevalie@loria.fr

Abstract. We present a procedure for deciding security of protocols employing the Exclusive or operator. This procedure relies on a direct combination of a constraint solver for security protocol with a unification algorithm for the exclusive-or theory. Hence compared to the previous ones it is much simpler and easily amenable to automation. The principle of the approach can be applied to other theories too.

Security protocol analysis has been intensively studied [22, 24, 28, 16, 19, 18] motivated by the threats on internet communications and their dramatic consequences. Recently many procedures have been proposed to decide insecurity of cryptographic protocols in the Dolev-Yao model w.r.t. a finite number of protocol sessions [1, 5, 17, 27, 23, 2]. Among the different approaches the symbolic ones [23, 12, 26, 4, 20] are based on reducing the problem to constraint solving in a term algebra. Constraint solving has proved to be quite effective on standard benchmarks [14] and also was able to discover new flaws on several protocols [13].

However while most formal analysis of security protocols abstracts from low-level properties, i.e., certain algebraic properties of encryption, such as the multiplicativity of RSA or the properties induced by chaining methods for block ciphers, many real attacks and protocol weaknesses rely on these properties. In particular for the XOR operator (which is frequently used in protocol design) Ryan and Schneider [29] give a simple attack on Bull's recursive authentication protocol: the protocol is used to distribute a connected chain of keys linking all the nodes from originator to the server, but if one key is compromised the others can be compromised too thanks to the property of XOR. Conversely, if XOR is considered as a free operator then, as shown by L. Paulson using the Isabelle prover [25], the protocol is secure. For attacks exploiting the XOR properties in the context of mobile communications see [7].

Therefore several attempts have been made to extend the protocol decision procedures to incorporate algebraic properties in the Dolev-Yao model and relax in that way the so-called *perfect encryption assumption* (i.e. 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). To our knowledge only two such procedures have been proposed for handling the properties of the XOR operator: [9] gives a decision procedure for XOR with an optimal NP complexity and [15] describes a constraint solving procedure for a more general class of protocols but with a higher complexity. Both procedures seem difficult to be turned into effective verification tools. The former is based on guessing terms of some height and may lead to combinatorial explosion. The latter has also hard complexity and is intricate.

In this paper we present a new procedure for deciding security of protocols employing the XOR operator. This procedure relies on a direct combination of a constraint solver for security protocol with a unification algorithm for the XOR theory. Hence compared to the previous ones it is much simpler and easily amenable to automation. The principle of the approach can be applied to other theories too. An advantage of our approach is that it reuses as much as possible results from unification theory and especially combination techniques, instead of reconstructing them in ad-hoc way for security constraints.

* This work was partially supported by IST AVISPA HTTP://WWW.AVISPA-PROJECT.ORG/ and RNTL Prouvé

Structure of the paper. In the first two sections, we compare the result presented with the similar ones and we provide an example illustrating the role of XOR in attacks. We then present the theoretical framework of our stduy, and our model of protocols (in Section 3) and of the hostile environment in Section 4. After this, we define the Simultaneous Construction Problems, and the translation from protocols to SCPs in Section 5, and transformations on SCPs in Sections 6 and 7. Finally, we show on an example how this system permits to automatically find an attack on a protocol in Section 8.

The detailled proofs are given in the companion research report available on the web [11].

1 Related work

The main result presented in this article is the decidability of the search for attacks on a protocol within a finite number of sessions and no bounds on message length. This result is very similar to the one presented by V. Shmatikov and H. Comon-Lundh in [15], the main difference being that the protocol we consider are only a subset of the protocol they take into account. The reason for this is that we impose some conditions on the occurrences of the variables in the rules of the protocol. However, these conditions are no real restrictions in practice.

To explain why, let us consider the reception of a message m (a ground term) by an agent. Upon receiving this message, the agent matches it with a pattern t, and assigns values to the variables of t according to the result of this matching. We define a protocol to be *deterministic* when given m and t, there is at most one possible assignment, *i.e.* at most one substitution τ such that $t\tau \equiv m$. As is noted in [30], not every deterministic protocol satisfies our restrictions. However, as was already noted in [9], and proved in [8] every *deterministic* protocol can be effectively transformed in a protocol that satisfies these restrictions. We have not considered this larger class of deterministic protocols in order to avoid the introduction of an additional ordering on the variables as is done in [1], this additional ordering being encoded within the order on messages.

To sum up, we consider the result presented in this article is equivalent for practical purposes to the one presented in [15]. But we claim that this article is very original because the algorithm presented and the proof of it are very different. In [15], but also in all recent papers considering the problem of cryptographic protocols analysis in presence of an algebraic operators [9, 10, 30, 6], the equalities induced by the theory of the operator are hard-wired into the deduction system and into the proof. The proofs are technically involved when only one algebraic operator is considered, and we do not know any attempt to consider several independent algebraic operators at the same time.

On the other hand, it can be shown that the system presented in this article terminates, is sound and is complete as soon as the theory has some simple properties. We plan to write soon an extension of the result presented to take into account several algebraic operators.

2 A Motivating Example

We illustrate that when taking the algebraic properties of XOR into account, new attacks can occur. As an example, we use a variant of the Needham-Schroeder-Lowe Protocol [21], i.e., the public-key Needham-Schroeder Procotol with Lowe's fix, where in some place, instead of concatenation XOR is used. Using common notation, the protocol is given as follows:

$$
\begin{aligned}
&1.\ A \rightarrow B:\ \{N_A, A\}^p_{K_B}\\
&2.\ B \rightarrow A:\ \{N_B, \oplus(\{N_A, B\})\}^p_{K_A}\\
&3.\ A \rightarrow B:\ \{N_B\}^p_{K_B}
\end{aligned}
$$

If XOR is interpreted as free symbol, such as pairing, then according to [21] this protocol is secure. In particular, the intruder is not able to get hold of N_B. However, if the algebraic properties of XOR are taken into account, the following attack is possible, which is a variant of the original attack on the Needham-Schroeder Protocol and which allows the intruder I to obtain N_B. In this attack, two sessions run interleaved where the steps of the second session are marked with $'$. In the first session, A talks to the intruder I, and in the second session I, purporting to be A, talks to B. We emphasize that in this attack I generates new messages by applying the XOR operator and uses that $N_A \oplus B \oplus I \oplus B = N_A \oplus I$.

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

3 Terms and Protocols

3.1 Messages and Knowledge

In this paper we assume the same setting as the one considered in [9]. The messages are modelled by terms over a signature containing a denumerable number of free constants and:

$$\mathcal{F} = \left\{ \langle \_, \_ \rangle, \{\_\}^p_\_, \_^{-1}, \{\_\}^s_\_, \_ \oplus \_, 0 \right\}$$

The $\langle \_, \_ \rangle$ represents the concatenation (pairing) of its two arguments.The constructor $\{\_\}^p_\_$ represents the public key encryption, and for a public key $k \in \mathcal{K}\mathrm{ey}$, the term k^{-1} represents the inverse key. The operator $\{\_\}^s_\_$ represents the symmetric key encryption. We call $\mathcal{F}_f$ the set of these operators. Under the hypothesis of non-collision between messages, the operators in $\mathcal{F}_f$ are *free*: For $f, g \in \mathcal{F}_f$, the equality $f(t_1, t_2) = g(t'_1, t'_2)$ holds if, and only if, $f = g$, $t_1 = t'_1$ and $t_2 = t'_2$.

On the other hand, the $\oplus$ operator represents the *Exclusive-Or* operation, and 0 is the constant representing sequences of the 0 bit-value of any length. Given two arbitrary messages a and b we consider that this operator has the following properties:

$$\begin{array}{lr}
x \oplus (y \oplus z) = (x \oplus y) \oplus z & (A) \\
x \oplus y = y \oplus x & (C) \\
x \oplus 0 = x & (U) \\
x \oplus x = 0 & (N)
\end{array}$$

If we orient from left to right the equations (U) and (N), we get a convergent rewrite system modulo AC. Given a term t, we note $\ulcorner t \urcorner$ the normal form of t. For example, we have:

- $\ulcorner \oplus(\{\oplus(\{a, b\}), b, c\}) \urcorner = \oplus(\{a, c\})$
- $\ulcorner \oplus(\{a, \oplus(\{b, c\})\}) \urcorner = \oplus(\{a, b, c\})$

Given a term t, we define the *factors* of t, denoted $\mathrm{Factor}(t)$:

$$\mathrm{Factor}(t) = \begin{cases} \{t_1, \ldots, t_n\} & \text{if } \ulcorner t \urcorner = \oplus(\{t_1, \ldots, t_n\}) \\ \{t\} & \text{otherwise} \end{cases}$$

We note that all the factors of a normalized term $\ulcorner t \urcorner$ have a free root operator.

The *subterms* are defined on normalized terms. Given a term t such that $t = \ulcorner t \urcorner$, the set of its subterms $\mathrm{Sub}(t)$ is defined inductively by:

$$\mathrm{Sub}(t) = \{t\} \cup \begin{cases} \mathrm{Sub}(t_1) \cup \mathrm{Sub}(t_2) & \text{If } t = f(t_1, t_2) \\ & \text{with } f \text{ a free symbol} \\ \cup_{u \in \mathrm{Factor}(t)} \mathrm{Sub}(u) & \text{Otherwise} \end{cases}$$

Given a set of terms E, we also note $\mathrm{Sub}(E)$ the set $\cup_{t \in E} \mathrm{Sub}(t)$.

If the root operator of a *normalized* term t is a free constructor or a constant, we say t is a free term. Otherwise, we say t is a $\oplus$ term.

Last, we call $\mathcal{X}$ the set of variables, and $\mathrm{T}(\mathcal{F}, \mathcal{X})$ (resp. $\ulcorner \mathrm{T}(\mathcal{F}, \mathcal{X}) \urcorner$) the set of terms (resp. normalized terms). *Substitutions*, noted σ, τ, $\ldots$ are defined as *mappings* from $\mathcal{X}$ to $\ulcorner \mathrm{T}(\mathcal{F}, \mathcal{X}) \urcorner$. The application of a substitution σ on a term t, denoted $t\sigma$, consists in the term where all variables $x_1, \ldots, x_n$ of t are replaced by the terms $x_1\sigma, \ldots, x_n\sigma$.

3.2 Protocols

In order to decide the security of a protocol w.r.t secrecy or authentication for a fixed number of protocol sessions can be reduced to deciding the security for a single session by guessing an interleaving of the sessions that leads to the security violation [27]. In the same way we can assume that the protocol steps are linearly ordered (otherwise we simply try all possible orderings). Hence we will consider here only a single session of a protocol defined as a sequence of steps.

The following definition is explained below.

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

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

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

Intuitively for executing a protocol step $R_\iota \Rightarrow S_\iota$ on receiving a (normalized) message m in a protocol run it is first checked whether m and R_ι match, i.e., whether there exists a ground substitution σ such that $m =_E R_\iota\sigma$. If so $\ulcorner S_\iota\sigma \urcorner$ is returned as output. We always assume that the messages exchanged between principals (and the intruder) are normalized — therefore, m is assumed to be normalized and the output of the above rule is not $S_\iota\sigma$ but $\ulcorner S_\iota\sigma \urcorner$. This is because principals and the intruder cannot distinguish between equivalent terms and therefore they may only work on normalized terms (representing the corresponding equivalence class of terms). Finally we note that since the different protocol rules may share variables, some of the variables in R_ι and S_ι may be already bounded by substitutions obtained from previous applications of 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.

Condition 1. , in the above definition is not a restriction since the transformation performed by a protocol rule and its normalized variant coincide. Condition 2. guarantees that when an output is produced with S_ι all variables in S_ι are already "bounded". 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 $\oplus(\{x, y\}) \Rightarrow \langle x, y \rangle$ is the first one and thus, x and y are not bounded, then this rule violates Condition 3: On receiving

$\oplus(\{a, b, c\})$, for instance, different substitutions are possible, including $\{x \mapsto \oplus(\{a, b\}), y \mapsto c\}$, $\{x \mapsto \oplus(\{b, d\}), y \mapsto \oplus(\{a, c, d\})\}$, etc. In other words, a principal must guess a substitution. We have shown in [8] that every deterministic protocol can be transformed in a protocol satisfying Condition 3.

The protocol informally described in Section 2 can formally be stated as follows: Agent a plays role A and agent b role B; We define $\mathcal{I} = \{1, 2, 3, 4\}$; The initial knowledge of the intruder is $\mathcal{S} = \big\{0, \mathrm{a}, b, I, ki, ki^{-1}, ka, kb\big\}$, and the protocol rules are:

$$\begin{array}{cccc} 1: & 0 & \Rightarrow & \{\langle na, \mathrm{a}\rangle\}^p_{ki} \\ 2: & \{\langle x_{na}, \mathrm{a}\rangle\}\, kb & \Rightarrow & \{\langle nb, \oplus(\{x_{na}, b\})\rangle\}^p_{ka} \\ 3: & \{\langle x_{nb}, \oplus(\{na, I\})\rangle\}^p_{ka} & \Rightarrow & \{x_{nb}\}^p_{ki} \\ 4: & \{nb\}^p_{kb} & \Rightarrow & 0 \end{array}$$

Our aim is to determine, given the initial knowledge of the intruder, the set of substitutions σ such that this ordering of messages corresponds to a possible execution. Before proceeding further, we need to formalize the deduction abilities of the intruder.

4 Threat Model

4.1 The Intruder

We assume that a protocol is run across a hostile environment where the source of the received messages cannot be established. This environment is modeled by an evil actor, called the *intruder*, trying to reach a state that should be banned by the protocol. If the goal of the protocol is to ensure the secrecy of a data M, the forbidden states are those where the intruder knows M. If the goal of the protocol is to ensure authentication of the participants, the forbidden states are those where a honest participant wrongly assumes that a message M it has received originates from another participant.

An attack on a trace-based property of a protocol can be viewed as a particular interleaving of a finite number of protocol sessions. For instance an attack on secrecy can be modeled by adding a message to the protocol where the intruder has to send the confidential piece of data. If this extended protocol admits a *feasible* execution then the initial protocol can be considered as insecure.

An execution is feasible if every message received by the honest agents can be derived by the intruder from his initial knowledge and intercepted messages. Hence building an insecure protocol execution reduces to a system of *constraints* to be solved in a particular term algebra. The variables to be solved correspond to the part of the protocol messages that are not read by the honest agents (e.g. because they are encrypted with an unknown key).

The security of the protocol is assessed versus an intruder as strong as possible. We assume it can *divert* all messages sent by honest participants and add their content to its knowledge, it can *send* messages its knowledge permits to deduce under the identity of other actors, and may perform deductions over its knowledge to this end.

4.2 Intruder Deduction Rules

The *knowledge* of the intruder is represented by a set of normalized messages, *i.e.* is a subset of $\ulcorner \mathrm{T}(\mathcal{F}, \mathcal{X}) \urcorner$. The deductions that the intruder can perform from its knowledge are modeled by rewrite rules $l \to r$ (read: From l deduce r) where l is a set of messages (a subset of $\mathrm{T}(\mathcal{F}, \mathcal{X})$) and r is a message (a term). The right-hand side of a rule is its *result*. We shall only consider rules $l \to r$ where both l and r are in normal form (*i.e.* $\ulcorner l \urcorner = l$ and $\ulcorner r \urcorner = r$). In order to have lighter notation, and under this hypothesis, we omit the normalization function $\ulcorner \_ \urcorner$ when not necessary. The available deduction rules are split into two disjoint sets: Composition rules and decomposition rules.

Let $l \to r$ be a deduction rule. It is a decomposition rule iff there exists $t \in l$ such that r is a strict maximal subterm of t. Otherwise, it is a composition rule. We always assume $r \notin l$, since such rules do not permit to deduce new terms.

In Tables 4.2 and 2 we give the rules considered in this paper. The rule:

$$a_1, \ldots, a_n \to \ulcorner \oplus(a_1, \ldots, a_n) \urcorner$$

is a composition rule if the right-hand side is a $\oplus$-term, and a decomposition rule if its head operator is a constant or a free constructor.

Name	Deduction rule
$\mathrm{L}_{\mathrm{c},\langle\_,\_\rangle}$	$a, b \to \langle a, b\rangle$
$\mathrm{L}_{\mathrm{c},\{\_\}^{\mathrm{s}}\_}$	$a, b \to \{a\}_b^s$
$\mathrm{L}_{\mathrm{c},\{\_\}^{\mathrm{p}}\_}$	$a, b \to \{a\}_b^p$
$\mathrm{L}_{\mathrm{c},\oplus}$	$a_1, \ldots, a_n \to \ulcorner \oplus(\{a_1, \ldots, a_n\}) \urcorner$

Table 1. Composition rules

Name	Deduction rule	Decomposed term	Condition
$\mathrm{L}_{\mathrm{d},\langle\_,\_\rangle}{}^1$	$\langle a, b\rangle \to a$	$\langle a, b\rangle$	$\emptyset$
$\mathrm{L}_{\mathrm{d},\langle\_,\_\rangle}{}^2$	$\langle a, b\rangle \to b$	$\langle a, b\rangle$	$\emptyset$
$\mathrm{L}_{\mathrm{d},\{\_\}^{\mathrm{s}}\_}$	$\{a\}_b^s, b \to a$	$\{a\}_b^s$	b
$\mathrm{L}_{\mathrm{d},\{\_\}^{\mathrm{p}}\_}$	$\{a\}_b^p, b^{-1} \to a$	$\{a\}_b^p$	b^{-1}
$\mathrm{L}_{\mathrm{d},\oplus}$	$a_1, \ldots, a_n \to \ulcorner \oplus(\{a_1, \ldots, a_n\}) \urcorner$	a_1	$\emptyset$

Table 2. Decomposition rules

We note L the set of all deduction rules of the intruder.

Transition relation and derivations. Let E and F be two normalized sets of terms, and L a subset of L. We write $E \to_L F$ if there exists a rule $l \to r$ in L such that $l \subseteq E$ and $F = \{r\} \cup E$. We denote by $\to^*$ the reflexive and transitive closure of $\to$. If $L = \{l \to r\}$, we simply note $E \to_{l \to r} F$. Without loss of generality, we also always assume that if $E \to_{l\to r} F$, then $r \notin E$. Under this assumption, a sequence of transitions $E_1 \to_L \ldots \to_L E_n$ is called a *derivation* on L. We say this derivation starts from E_1 and has goal $E_n \setminus E_{n-1}$. A derivation $D : E \to_L^* F$ starting from E and of goal t is defined to be *well-formed* if $F \subseteq \mathrm{Sub}(E \cup \{t\})$.

We have proved in [9] the following useful result:

Proposition 1 *(Existence of well-formed derivations) Let E be a normalized set of terms and t be a term in normal form. There exists a well-formed formed derivation on* L *starting from E of goal t if, and only if, there exists a derivation on* L *starting from E of goal t.*

4.3 Set of Deducible Messages and Properties

Let R be any system of rewrite rules over sets of terms.

Definition 2 *We note $\overline{E}^R$ the set of messages deducible from E using the rewrite system R:*

$$\overline{E}^R = \{t \mid \exists E', E \rightarrow_R^* E' \text{ and } t \in E'\}$$

In the case of intruder deduction system L, we simply note $\overline{E}$ the set $\overline{E}^{\mathrm{L}}$. We have given the proofs of the two following propositions as they are both easy and very important. Indeed, Proposition 3 permits to link a set of messages with all the possible instantiations of the variables.

Proposition 2 *Suppose $F \subseteq \overline{E}$. Then $\overline{E} = \overline{F \cup E}$*

PROOF. This proposition is based on the fact that the $\overline{}$ operator is idempotent ($\overline{\overline{E}} = \overline{E}$) and growing for $\subseteq$ ($E \subseteq F$ implies $\overline{E} \subseteq \overline{F}$). The announced equality follows from these two properties by a double inclusion argument.

Proposition 3 *Let E and F be two sets of terms. We have $\overline{E} = \overline{F}$ iff for all substitutions σ, we have $\overline{E\sigma} = \overline{F\sigma}$*

PROOF. The right to left direction is trivial: Consider the Identity substitution. To prove the left to right direction, we note that for all sets of terms E, if $E \rightarrow_{\mathrm{L}}^* E'$, then $\overline{E} = \overline{E'}$. The equality $\overline{E} = \overline{F}$ implies there exists a set of terms G such that $E \rightarrow^* G$ and $F \rightarrow^* G$. Let σ be a substitution. One can check that $l \rightarrow r \in \mathrm{L}$ implies $\ulcorner l\sigma \urcorner \rightarrow \ulcorner r\sigma \urcorner \in \mathrm{L}$. Thus, we have $\ulcorner E\sigma \urcorner \rightarrow^* \ulcorner G\sigma \urcorner$ and $\ulcorner F\sigma \urcorner \rightarrow^* \ulcorner G\sigma \urcorner$. By construction, we have:

$$\begin{cases} \ulcorner G\sigma \urcorner \subseteq \overline{\ulcorner E\sigma \urcorner} \\ \ulcorner G\sigma \urcorner \subseteq \overline{\ulcorner F\sigma \urcorner} \end{cases}$$

Thus, Proposition 2 permits to conclude.

5 Simultaneous Construction Problems

We now introduce constraints that an intruder has to solve in order to build a protocol execution leading to an attack.

Definition 3 *A* Construction Problem *is a pair (E, t) noted $E \rhd t$ with E a finite subset of $\ulcorner \mathrm{T}(\mathcal{F}, \mathcal{X}) \urcorner$ and $t \in \ulcorner \mathrm{T}(\mathcal{F}, \mathcal{X}) \urcorner$. A substitution σ satisfies $E \rhd t$ iff $\ulcorner t\sigma \urcorner \in \overline{\ulcorner E\sigma \urcorner}$. In this case, we note:*

$$\sigma \models E \rhd t$$

Definition 4 *A* Simultaneous Construction Problem (SCP) *is a finite sequence $(E_i \rhd t_i)_{i \in \{1,\ldots,n\}}$ of construction problems. such that:*

1. *for all $i \in \{1, \ldots, n\}$ and for all $x \in \mathrm{Var}(E_i)$, there exists $j < i$ such that $x \in \mathrm{Var}(t_j)$;*
2. *for all $i, j \in \{1, \ldots, n\}$ with $i < j$, there exists $F_j \subseteq E_j$ such that $\overline{F_j} = \overline{E_i}$*
3. *for all $i \in \{1, \ldots, n\}$, if $\oplus(\{u_1, \ldots, u_l\}) \in \mathrm{Sub}(t_i)$, then there exists at most one j such that $\mathrm{Var}(u_j) \not\subseteq \mathrm{Var}(t_1, \ldots, t_{i-1})$.*

Moreover, we assume that if $n \geq 1$, then $E_1 \neq \emptyset$. In the case of protocol analysis, this can be ensured *e.g.* by stating that the intruder always knows her name. The last condition is true by the Condition 3. on protocols when a SCP is built by an execution order over a protocol P. It will be easy to see that all transformations on SCP's that will be defined in the following preserve this property.

Definition 5 *(Satisfiability of SCP) Let L be a set of deduction rules over sets of terms. A SCP $\mathcal{C}$ is σ-satisfiable for L if:*

$$\text{for all } E \rhd t \text{ in } \mathcal{C}, \quad \sigma \models E \rhd_L t$$

We note $\mathsf{Sol}(\mathcal{C})$ the set of substitutions σ such that $\mathcal{C}$ is σ-satisfiable.

The next definition will allow us to obtain a generic description of the set of solutions of an SCP. Since attacks in our setting are substitutions let us define the set of prefixes of a set of substitutions.

Definition 6 *(Prefix set) A set of substitutions Θ is a* prefix *of a set of substitutions Σ if:*

$$\begin{cases} \forall \sigma \in \Sigma \ \exists \tau \ \exists \rho \in \Theta, \sigma = \rho\tau \\ \forall \tau \in \Theta \ \exists \rho \ \tau\rho \in \Sigma \end{cases}$$

The first condition ensures that all substitutions in the set Σ are instances of a substitution in the prefix set. The second condition ensures that each substitution in the prefix set covers a non-empty subset of Σ. Thus, if Θ is a prefix set of Σ, the set Θ is empty if, and only if, Σ is empty. Note that the notion of prefix set is weaker than the notion of most general unifier (summarized as: All the instances are solutions). Suppose Θ is the prefix set of a set Σ, and let $\tau \in \Theta$. Then there might exist a substitution ρ such that $\tau\rho$ is *not* in Σ. For example, the set $\{\mathrm{Id}\}$ is a prefix set of any non-empty set of substitutions Σ but unless Σ is the set of all substitutions, not all instances of Id are in Σ

Connection with Protocols. Let $P = (\{R_\iota \Rightarrow S_\iota,\ \iota \in \mathcal{I}\}, \mathcal{S})$ be a protocol with $\mathcal{I} = \{1, \ldots, n\}$. A sequence of ground terms $m_1, \ldots, m_n$ forms a valid trace of the protocol if there exists a substitution σ such that $m_i = \ulcorner S_i\sigma \urcorner$ for all $i \in \mathcal{I}$ and if the intruder was able at every stage to deduce $\ulcorner R_i\sigma \urcorner$. This condition can be formalized as follows. Let F_i be the knowledge of the intruder after she has diverted the i-th message. One has:

$$\begin{cases} F_0 = \mathcal{S} \\ F_i = F_{i-1} \cup \left\{ \ulcorner S_i\sigma \urcorner \right\} \end{cases}$$

The intruder can deduce, at every stage, all m_i if, for all $i \in \mathcal{I}$, one has:

$$\ulcorner R_i\sigma \urcorner \in \overline{F_{i-1}}$$

Conversely, if these facts hold for σ, then $\ulcorner S_1\sigma \urcorner, \ldots, \ulcorner S_n\sigma \urcorner$ is a valid trace of the protocol.

Let us now define:

$$\begin{cases} E_0 = \mathcal{S} \\ E_i = E_{i-1} \cup \{S_i\} \end{cases}$$

From what precedes, all the valid traces of the protocol P are given by the solutions of the SCP:

$$\mathcal{C} = E_0 \rhd R_1, \ldots, E_{n-1} \rhd R_n$$

However, the set $\mathsf{Sol}(\mathcal{C})$ may be infinite. In order to decide whether a protocol has a secrecy attack, it is sufficient to be able to decide whether the SCP $\mathcal{C}$ associated to a protocol is satisfiable. But other trace-based properties need a finer result to be decided. For example, one needs to know the possible values of the protocol variables in order to decide if a given protocol has an authentication flaw. Thus, our aim is not to decide the emptyness of the set $\mathcal{C}$, but to find a finite and symbolic representation of it. As a consequence, we will solve the following more general problem:

– $\textsc{Prefix}(\mathcal{C})$: find a finite prefix set Θ of $\mathsf{Sol}(\mathcal{C})$;

5.1 Solved form

We first introduce the notion of SCP in *solved form*. This notion is a generalization of the one considered in [23] for the Dolev-Yao intruder in the free theory. Given a set of variables $\mathcal{X}$ let us note $\mathrm{FV}(t) = \mathrm{Factor}(t) \cap \mathcal{X}$.

Definition 7 *(Solved form) Let $\mathcal{C} = (E_i \rhd t_i)_{i \in \{1,\ldots,n\}}$ be a SCP, and let $\mathcal{X}$ be the set of variables appearing in $\mathcal{C}$. We say $\mathcal{C}$ is in* solved form *if:*

- $\forall i \in \{1, \ldots, n\}, \mathrm{FV}(t_i) = \{x_i\}$ *and* $i \neq j$ *implies* $x_i \neq x_j$*;*
- $\mathcal{X} = \{x_1, \ldots, x_n\}$.

This notion is crucial because of the following proposition.

Proposition 4 *Every SCP in solved form is satisfiable.*

PROOF. Let $\mathcal{C} = (E_i \rhd t_i)_{i \in \{1,\ldots,n\}}$ be a SCP in solved form, and let $\mathcal{X} = \{x_1, \ldots, x_n\}$. The conditions on the solved form imply that there exists a bijection between the construction equations $E_i \rhd t_i$ and the variables. Hence up to re-indexing if necessary, one has either $t_i = x_i$ or $t_i = \oplus(\{x_i, t_{i,1}, \ldots, t_{i,n_i}\})$. The third property of SCPs then implies that for all $i \in \{1, \ldots, n\}$, one has $\mathrm{Var}(t_{i,1}, \ldots, t_{i,n_i}) \subseteq \mathrm{Var}(t_1, \ldots, t_{i-1})$. For $i \in \{1, \ldots, n\}$, let $u_i = 0$ if $t_i = x_i$ and $u_i = \oplus(\{t_{i,1}, \ldots, t_{i,n_i}\})$ otherwise. The unification problem $\mathcal{U}$:

$$\forall i \in \{1, \ldots, n\}, \quad x_i \stackrel{?}{=} u_i$$

is then in solved form: One obtains a solution σ by replacing, for i from 1 to n, all the variables y in u_i by $y\sigma$ which is already computed. By definition of σ as a solution of $\mathcal{U}$, we have:

$$\ulcorner \mathcal{C}\sigma \urcorner = \ulcorner E_1\sigma \urcorner \rhd 0, \ldots, \ulcorner E_n\sigma \urcorner \rhd 0$$

which is trivially satisfiable. Hence $\mathcal{C}$ has at least one solution σ.

6 Normalization of SCP's

We introduce a *normalization* procedure for transforming an SCP into a simpler equivalent one. In the remaining part of this subsection, we give some normalization rules on SCP's. In this paper we only give normalization rules to prove the completeness of $\mathcal{L}$. For a faster constraint-solving algorithm one may add more normalization rules as the ones given in [8] for free operators such as encryption and pairing.

Proposition 5 *Let*

$$\mathcal{C} = \mathcal{C}_\alpha, E \cup \{u_x\} \rhd t, \mathcal{C}_\beta$$

be a SCP such that there exists $E_x \rhd t_x$ in $\mathcal{C}_\alpha$ with $\mathrm{FV}(t_x) = \mathrm{FV}(u_x) = \{x\}$. Then $\mathcal{C}$ is equivalent to:

$$\mathcal{C}' = \mathcal{C}_\alpha, E \cup \left\{\ulcorner \oplus(\{u_x, t_x\}) \urcorner\right\} \rhd t, \mathcal{C}_\beta$$

We note $\mathcal{C} \Rightarrow_{\oplus,l} \mathcal{C}'$.

Proposition 6 *Let*

$$\mathcal{C} = \mathcal{C}_\alpha, E \rhd \{u_x\}, \mathcal{C}_\beta$$

be a SCP such that there exists $E_x \rhd t_x$ in $\mathcal{C}_\alpha$ with $\mathrm{FV}(t_x) = \mathrm{FV}(u_x) = \{x\}$. Then $\mathcal{C}$ is equivalent to:

$$\mathcal{C}' = \mathcal{C}_\alpha, E \rhd \left\{\ulcorner \oplus(\{u_x, t_x\}) \urcorner\right\}, \mathcal{C}_\beta$$

We note $\mathcal{C} \Rightarrow_{\oplus,r} \mathcal{C}'$.

Note that in Propositions 5 and 6, by definition of the normalization function $\ulcorner\_\urcorner$, we have $\mathrm{FV}(\ulcorner\oplus(\{u_x, t_x\})\urcorner) = \emptyset$.

Proposition 7 *For all rewrite system L, we have $\overline{E}^L = \overline{F}^L$ iff for all substitutions σ, we have $\overline{E\sigma}^L = \overline{F\sigma}^L$*

We write $\mathcal{C} \Rightarrow \mathcal{C}'$ if either $\mathcal{C} \Rightarrow_{\oplus,r} \mathcal{C}'$ or $\mathcal{C} \Rightarrow_{\oplus,l} \mathcal{C}'$. A SCP $\mathcal{C}$ is in normal form if there does not exist $\mathcal{C}'$ such that $\mathcal{C} \Rightarrow \mathcal{C}'$ and if all its subterms are in normal form for $\ulcorner\_\urcorner$.

7 A System for Solving SCP

We transform the SCP using rules of a system $\mathcal{L}$. These rules can be partitionned in two subsets: those applying to terms whose root symbol, for a solution σ, is in the free theory, called $\mathcal{L}_{\mathrm{f}}$, and those applying to xor terms, called $\mathcal{L}_{\oplus}$. We now define these systems.

The notation $\sigma \in \mathrm{mgu}\,(u, t)$ means that the substitution σ is a most general solution of the unification problem $u \stackrel{?}{=} t$ over the theory $E = E_1 \cup E_2$, with E_1 the free theory over standard Dolev-Yao operations (concatenation $-\langle\_,\_\rangle-$, symmetric key encryption $-\{\_\}^s_\_-$ and public key encryption $-\{\_\}^p_\_-$), and E_2 is the ACUN theory of the $\oplus$ operator and 0. Note that the set of most general unifiers is either empty or finite up to renaming.

7.1 The Rules in $\mathcal{L}_{\mathrm{f}}$

In Figure 1 the symbol f is any free operator in $\{\langle\_,\_\rangle, \{\_\}^s_\_, \{\_\}^p_\_\}$. Intuitively, the Comp rule is the counterpart for the SCPs of the rules of Table 4.2 that are associated with free constructors. Symmetrically, the Dec rule is the counterpart of the rules of Table 2 associated with free constructors.

The Unif rule plays a different role. It is applied when the term to be built is in the knowledge of the intruder once a substitution is applied.

$$\text{Comp}: \quad \frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, E \rhd x_1, E \rhd x_2, \mathcal{C}_\beta)\sigma} \ \sigma \in \mathrm{mgu}\,(t, f(x_1, x_2))$$

$$\text{Dec}: \quad \frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, E \rhd \mathrm{Cond}(f(x_1, x_2)), E \cup \mathrm{Res}(f(x_1, x_2)) \rhd t, \mathcal{C}_\alpha)\sigma} \ u \in E,\ \sigma \in \mathrm{mgu}\,(u, f(x_1, x_2))$$

$$\text{Unif}: \quad \frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, \mathcal{C}_\beta)\sigma} \ u \in E,\ \sigma \in \mathrm{mgu}\,(u, t)$$

Fig. 1. System $\mathcal{L}_{\mathrm{f}}$ of transformation rules.

7.2 The Rules in $\mathcal{L}_{\oplus}$

The rules in $\mathcal{L}_{\oplus}$ are used to treat specifically the case of the xor operator. They are given in Figure 2.

The $\mathsf{Comp}_{\oplus}$ (resp. $\mathsf{Dec}_{\oplus}$) rule has no real counterpart in Tables 4.2 and 2. On the other hand, the $\mathsf{Apply}_{\oplus}$ rule corresponds to the application of a deduction rule:

$$a_1, \ldots, a_n \rightarrow \ulcorner\oplus(\{a_1, \ldots, a_n\})\urcorner$$

$$\mathsf{Comp}_{\oplus}: \quad \frac{\mathcal{C}_\alpha, E \rhd \oplus(\{t_1,\ldots,t_n\}), \mathcal{C}_\beta}{\mathcal{C}_\alpha, E \rhd t_1, E \rhd \oplus(\{t_2,\ldots,t_n\}), \mathcal{C}_\beta}$$

$$\mathsf{Dec}_{\oplus}: \quad \frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{\mathcal{C}_\alpha, E \rhd \oplus(\{t_2,\ldots,t_n\}), E \cup \{t_1\} \rhd t, \mathcal{C}_\beta} \; \oplus(\{t_1,\ldots,t_n\}) \in E$$

$$\mathsf{Apply}_{\oplus}: \quad \frac{\mathcal{C}_\alpha, E \rhd t, \mathcal{C}_\beta}{(\mathcal{C}_\alpha, \mathcal{C}_\beta)\sigma} \; U \subseteq E,\ \sigma \in \mathrm{mgu}\,(\textstyle\bigoplus_{u\in U} u, t)$$

Fig. 2. System $\mathcal{L}_\oplus$ of transformation rules.

regardless of whether this is a composition or decomposition rule.

We note $\mathcal{C} \rightarrow_l \mathcal{C}'$ if l is an applicable rule of $\mathcal{L}$.

7.3 Restrictions on the Application of Rules in $\mathcal{L}$

Consider a rule of $\mathcal{L}$:

$$\frac{\mathcal{C} = \mathcal{C}_\alpha, Eq, \mathcal{C}_\beta}{\mathcal{C}'}$$

Note that in the above notation, $\mathcal{C}_\alpha$ is a SCP, but not $\mathcal{C}_\beta$. We do several hypotheses on $\mathcal{C}$ and $\mathcal{C}$' for such a rule to be applicable.

First, we assume the SCP $\mathcal{C}$ is normalized. This means that, when applying rules, one should start from a SCP in normal form, and each SCP deduced must be normalized before being employed in further deductions.

Second, we impose in the above rule that $\mathcal{C}_\alpha$ has to be empty or in solved form. Moreover, we impose that if a rule applies on a term u or t, then:

$$\mathrm{FV}(u) = \mathrm{FV}(t) = \emptyset$$

Since the SCP is normalized, this is always the case for a term on the left-hand side of a construction problem. If the rule applies on the right-hand side t of $E \rhd t$, this restriction implies this problem is not in solved form.

In order to ensure termination, we also forbid the application of the Dec rule twice on the same term. This may be formalized using *e.g.* tagging of terms.

7.4 Decidability of $\mathrm{Prefix}(\mathcal{C})$

We prove in [11] that the transformation system presented, together with the normalization rules, permits to decide the problem $\mathrm{Prefix}(\mathcal{C})$. Starting from $\mathcal{C}$, rules of $\mathcal{L}$ are applied to generate as many SCPs as possible. For each generated SCP $\mathcal{C}$', let $\sigma_{\mathcal{C}'}$ be the substitution applied along the deductions starting from $\mathcal{C}$ and ending in $\mathcal{C}$'. Finally, let us define:

$$\Pi_{\mathcal{C}} = \{\sigma_{\mathcal{C}'} \,|\, \mathcal{C}' \text{ in solved form}\}$$

We prove:

1. Termination: The procedure terminates, and thus $\Pi_{\mathcal{C}}$ is finite;

2. Correctness: Every $\sigma \in \Pi_{\mathcal{C}}$ is a prefix of a substitution in $\mathsf{Sol}(\mathcal{C})$;
3. Completeness: For every $\tau \in \mathsf{Sol}(\mathcal{C})$, there exists $\sigma \in \Pi_{\mathcal{C}}$ such that σ is a prefix of τ.

Thus, $\Pi_{\mathcal{C}}$ is a prefix set of $\mathsf{Sol}(\mathcal{C})$ and we can state the following theorem:

Theorem 1 *The problem* $\mathrm{Prefix}(\mathcal{C})$ *is decidable.*

8 Example

We now illustrate our procedure on the attack example of Section 2. Let us note $E_0 = \left\{a, b, ka, kb, I, ki, ki^{-1}\right\}$ the initial knowledge of the intruder and:

- $E_1 = E_0 \cup \{\{\langle na, a\rangle\}^p_{ki}\}$
- $E_2 = E_1 \cup \{\{\langle nb, \oplus(\{x_{na}, b\})\rangle\}^p_{ka}\}$
- $E_3 = E_2 \cup \{\{x_{nb}\}^p_{ki}\}$

Given the protocol description in Section 3, the SCP corresponding to the search of an attack is:

$$E_0 \rhd 0, E_1 \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,,\\ E_2 \rhd \{\langle x_{nb}, \oplus(\{na, I\})\rangle\}^p_{ka}\,, E_3 \rhd \{nb\}^p_{kb}$$

The first construction problem can be eliminated by the Unif rule. In order to simplify notations, let $\mathcal{C}_\beta = E_2 \rhd \{\langle x_{nb}, \oplus(\{na, I\})\rangle\}^p_{ka}\,, E_3 \rhd \{nb\}^p_{kb}$.The intruder immediately decompose the message sent by a.

$$\cfrac{\cfrac{\cfrac{E_1 \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,, \mathcal{C}_\beta}{E_1 \rhd ki^{-1}, E_1 \cup \{\langle na, a\rangle\} \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,, \mathcal{C}_\beta}\ \mathsf{Dec}}{E_1 \cup \{\langle na, a\rangle\} \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,, \mathcal{C}_\beta}\ \mathsf{Unif}}{E_1 \cup \{na, a, \langle na, a\rangle\} \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,, \mathcal{C}_\beta}\ \mathsf{Dec}$$

From now on, we note $E_1' = E_1 \cup \{na, a, \langle na, a\rangle\}$. We apply twice the Comp rule on the first remaining problem:

$$\cfrac{\cfrac{E_1' \rhd \{\langle x_{na}, a\rangle\}^p_{kb}\,, \mathcal{C}_\beta}{E_1' \rhd \langle x_{na}, a\rangle\,, E_1' \rhd kb, \mathcal{C}_\beta}\ \mathsf{Comp}}{E_1' \rhd x_{na}, E_1' \rhd a, E_1' \rhd kb, \mathcal{C}_\beta}\ \mathsf{Comp}$$

The Unif rule again permits to eliminate the second and third construction problem since a and kb are in E_0. Note that up to this point, all the rules applied preserve the satisfiability, and thus could be implemented as normalization rules. The next step is non-deterministic, since the intruder does not know ka^{-1}.

Let σ be the most general unifier of the two terms $\{\langle x_{nb}, \oplus(\{na, I\})\rangle\}^p_{ka}$ and $\{\langle nb, \oplus(\{x_{na}, b\})\rangle\}^p_{ka}$. A simple calculation shows that σ is given by:

$$\sigma \ : \ \begin{cases} x_{nb} \mapsto nb \\ x_{na} \mapsto \oplus(\{na, b, I\}) \end{cases}$$

Let us note $E_3' = E_3\sigma$. We have:

$$E_3' = E_1 \cup \{\{\langle nb, \oplus(\{na, I\})\rangle\}^p_{ka}\,, \{nb\}^p_{ki}\}$$

The rule inference is:

$$\frac{E_1' \rhd x_{na}, E_2 \rhd \{\langle x_{nb}, \oplus(\{na, I\})\rangle\}^p_{ka}, E_3 \rhd \{nb\}^p_{kb}}{E_1' \rhd \oplus(\{na, b, I\}), E_3' \rhd \{nb\}^p_{kb}} \textsf{ Unif}$$

The $\textsf{Apply}_\oplus$ rule can be applied (with the identity substitution) to remove the first constraint:

$$\frac{b, I, na, \ldots \rhd \oplus(\{na, b, I\}), E_3 \rhd \{nb\}^p_{kb}}{E_3' \rhd \{nb\}^p_{kb}} \textsf{ Apply}_\oplus$$

Before proceeding further, the intruder decomposes the message $\{nb\}^p_{ki}$:

$$\cfrac{\cfrac{E_3' \rhd \{nb\}^p_{kb}}{E_3' \rhd ki^{-1}, E_3' \cup \{nb\} \rhd \{nb\}^p_{kb}} \textsf{ Dec}}{E_3' \cup \{nb\} \rhd \{nb\}^p_{kb}} \textsf{ Unif}$$

Finally, this last constraint is eliminated through the application of the rule $\textsf{Comp}$ followed by two applications of the rule $\textsf{Unif}$ with the identity substitution.

$$\cfrac{\cfrac{\cfrac{E_3' \cup \{nb\} \rhd \{nb\}^p_{kb}}{E_3' \cup \{nb\} \rhd nb, E_3' \cup \{nb\} \rhd kb} \textsf{ Comp}}{E_3' \cup \{nb\} \rhd nb} \textsf{ Unif}}{\emptyset} \textsf{ Unif}$$

In this case, the substitution found was:

$$\sigma \ : \ \begin{cases} x_{nb} \mapsto nb \\ x_{na} \mapsto \oplus(\{na, b, I\}) \end{cases}$$

Note that in this case, the substitution is ground. As a consequence, and by definition of solved forms, this implies that the final SCP produced is empty. Unless other sequences of transformations leading to different substitutions are found, this means that there is only one feasible execution of this protocol.

9 Conclusion

For sake of clarity we have not considered the issue of authentication flaws detection. This goal can be modelled by disequations between parts of messages sent and received. These disequations state that either the sender is not the one that is expected or the value received is different from the sent one. Note that the decidability result of [9] does not work with disequations and therefore cannot be easily extended to authentication properties. However the procedure we have presented in this paper computes a finite prefix set of all substitutions satisfying an execution order π. Applying the substitutions in the prefix set on the inequalities it should be easy to deduce whether some of them are satisfiable and thus if there exists an authentication flaw.

As future works we plan to finish the implementation of the protocol analysis procedure and tune it by exploiting possible optimizations of the unification algorithms as they were proposed by [3]. We shall also investigate other theories (e.g. abelian groups) to which the approach applies trying to find a general criteria.

References

1. R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proceedings of CONCUR'00*, volume 1877 of *Lecture Notes in Computer Science*, 2000.
2. A. Armando and L. Compagna. Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning. In *Foundation of Computer Security & Verification Workshops*, Copenhagen, Denmark, July 25-26 2002.
3. F. Baader and K. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 50–65. Springer-Verlag, 1992.
4. D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors, *Proceedings of ESORICS'03*, LNCS 2808, pages 253–270. Springer-Verlag, 2003. Available at `http://www.avispa-project.org`.
5. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th International Conference on Automata, Language and Programming: ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.
6. M. Boreale and M. G; Buscemi. Symbolic analysis of crypto-protocols based on modular exponentiation. In *Proceedings of the Mathematical Foundations of Computer Science 2003, 28th International Symposium (MFCS 2003)*.
7. N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of MOBICOM 2001*, pages 180–189, 2001.
8. Y. Chevalier. *Résolution de problèmes d'accessibilité pour la compilation et la validation de protocoles cryptographiques*. PhD thesis, LORIA, Université Henri Poincaré Nancy I, Vandoeuvre-les-Nancy, France, December 2003.
9. Y. Chevalier, R. Kuesters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Logic In Computer Science Conference LICS'03*, June 2003. Long version available as Technical Report RR-4697, INRIA, France.
10. 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, FSTTCS'03*, Lecture Notes in Computer Science. Springer, December 2003. Long version available as Christian-Albrecht Universität IFI-Report 0305, Kiel (Germany).
11. Y. Chevalier, M. Rusinowitch, M. Turuani, and L. Vigneron. A simple constraint-solving decision procedure for protocols with exclusive or. Research Report 5224, INRIA, 2004. `http://www.inria.fr/rrrt/liste-2004.html`.
12. Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of the Automated Software Engineering Conference (ASE'01)*. IEEE Computer Society Press, 2001. Long version available as Technical Report A01-R-140, LORIA, Nancy (France).
13. 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.
14. J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. Available via `http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz`, 1997.
15. H. Comon-Lundh and V. Shmatikov. Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or. In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 271–280, 2003.
16. Grit Denker, Jonathan Millen, and Harald Rueß. The CAPSL integrated protocol environment. Technical report, SRI International, October 2000.
17. M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.
18. Pierre Ganty Giorgio Delzanno. Automatic verification of time sensitive cryptographic protocols. In *TACAS*, LNCS, pages 342–356. Springer-Verlag, 2004.
19. Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In Dominique Méry Beverly Sanders, editor, *Fifth International Workshop on Formal Methods for Parallel Programming: Theory and Applications (FMPPTA 2000)*, number 1800 in Lecture Notes in Computer Science. Springer-Verlag, 2000. `http://www.dyade.fr/fr/actions/vip/jgl/cpv.ps.gz`.

20. Y. Lakhnech L. Bozga and M. Perin. Hermes: An automatic tool for the verification of secrecy in security protocols. In *Proceedings of the Computer-Aided Verification Conference CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 219–222. Springer-Verlag, 2003.
21. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.
22. Catherine Meadows. The nrl protocol analyzer: an overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
23. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.
24. J.C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of ssl 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, 1998.
25. L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, 1997.
26. Sandro Etalle Ricardo Corin. An improved constraint-based system for the verification of security protocols. In *SAS*, LNCS, pages 326–341. Springer-Verlag, 2002.
27. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.
28. Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
29. P.Y.A. Ryan and S.A. Schneider. An attack on a recursive authentication protocol. *Information Processing Letters*, 65, 1998.
30. V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proceedings of thirteenth European Symposium on Programming ESOP'04*, volume 2986 of *Lecture Notes in Computer Science*, pages 355–369,. Springer-Verlag, 2004.