On the Automatic Analysis of Recursive Security Protocols with XOR*

Ralf Küsters[1] and Tomasz Truderung[2]

[1] ETH Zurich
ralf.kuesters@inf.ethz.ch
[2] University of Kiel and Wrocław University
tomasz.truderung@ii.uni.wroc.pl

Abstract. In many security protocols, such as group protocols, principals have to perform iterative or recursive computations. We call such protocols *recursive protocols*. Recently, first results on the decidability of the security of such protocols have been obtained. While recursive protocols often employ operators with algebraic, security relevant properties, such as the exclusive OR (XOR), the existing decision procedures, however, cannot deal with such operators and their properties. In this paper, we show that the security of recursive protocols with XOR is decidable (w.r.t. a bounded number of sessions) for a class of protocols in which recursive computations of principals are modeled by certain Horn theories. Interestingly, this result can be obtained by a reduction to the case without XOR. We also show that relaxing certain assumptions of our model lead to undecidability.

1 Introduction

In group protocols and other classes of security protocols a protocol step performed by a principal (i.e., receiving a message and then sending a message) typically involves recursive or iterative computation. We will refer to such protocols by *recursive protocols*, in contrast to *non-recursive protocols* where the computation performed in one protocol step is simple and does not require recursion. Many, in fact, most of the recursive protocols proposed in the literature employ operators, such as Diffie-Hellman exponentiation and exclusive OR (XOR), which have algebraic, security relevant properties (see, e.g., [12, 13, 7]). The present work is concerned with the automatic security analysis of such protocols. While recently first results on the decidability of the security (more precisely, the secrecy property) of recursive protocols have been obtained [8, 10, 15], these results do not take into account operators with algebraic properties (see also the related work).

The attacks on recursive protocols presented in the literature illustrate that dealing with algebraic properties of operators is security relevant (see, e.g., [12, 13, 7]). One example is the Recursive Authentication (RA) protocol proposed by

* This work was partially supported by the DFG under grant KU 1434/4-1.

Bull and Otway [1]. In this protocol, a key distribution server receives a list of (arbitrary many) requests of pairs of principals who want to establish session keys among them. The server processes this list iteratively, generates the session keys, and then distributes them. In [11], Paulson proved that the RA protocol is secure under the assumption that session keys are distributed using (ideally) secure encryption. However, Ryan and Schneider [13] showed that there is an attack on the protocol if XOR is used to distribute keys, which in fact was the original proposition by Bull and Otway: In the attack by Ryan and Schneider, the adversary is given one session key generated by the server and using this key he can obtain all other session keys by chaining messages via XOR.

Contribution of this work. In this paper, we extend the model for recursive protocols proposed in [15], henceforth called the *Horn theory model*, by adding XOR (along with its algebraic properties). In the Horn theory model, recursive/iterative computations performed by principals in one protocol step are modeled by certain Horn theories, hence the name. While in models for non-recursive protocols XOR can be added without losing decidability of security (w.r.t. a bounded number of sessions) [2, 6]—security even remains NP-complete just as in the case without XOR [2]—for recursive protocols things are more involved. We show that a naïve extension of the Horn theory model by XOR leads to undecidability of security. (As a byproduct we also obtain undecidability in case complex keys are used in the Horn theory model, a fact that has not been observed before.) More precisely, we obtain undecidability in case principals may conjoin arbitrary messages received from the network by XOR. Conversely, we show decidability in case principals may only conjoin a fixed message, i.e., a message that does not depend on messages received from the network, with a message that depends on messages received from the network. We call protocols which only contain such principals $\oplus$-linear. From a practical point of view, $\oplus$-linear protocols are sufficient in many cases, e.g., for the RA protocol and other protocols [13, 2, 7]. We emphasize that we do not constrain the intruder in its ability to conjoin messages by XOR.

The technique used to obtain decidability is very different to the one in [15]. In fact, the main part of our proof is to reduce the security problem in the Horn theory model with XOR to the one without XOR. More precisely, we first prove certain properties of attacks involving XOR. Based on these properties we then reduce the security problem to the case without XOR, which by [15] is decidable. In the reduction we use the ability of principals to perform recursive computations in order to mimic applications of the XOR operator.

Further Related Work. For non-recursive protocols decidability of security (w.r.t. a bounded number of sessions) was shown for several operators with algebraic properties, e.g., XOR [2, 6], Diffie-Hellman Exponentiation [3, 14], and commuting public-key encryption [4]. However, the models employed in these works cannot handle recursive computations of principals, such as the computation of the server in the RA protocol. The techniques differ as well: Due to the absence of recursive computations, the reduction technique developed in the present work is not applicable in these models. Conversely, the techniques for

bounding the size of attacks and the constraint solving techniques employed in [2, 6, 3, 14, 4] cannot immediately be applied to recursive protocols.

In [8, 10], transducers were used to model recursive computations of principals. The expressiveness of these transducer-based models is orthogonal to the Horn theory model: While the transducer-based models allow to output messages of complex structure, in the Horn theory model only lists (or sets) of messages of a more simple structure can be produced. The main disadvantage of the transducer-based model is that, unlike the Horn theory model, messages cannot be tested for equality without losing decidability. This, as already observed in [10], immediately implies that security is undecidable in the transducer-based model with XOR (or Diffie-Hellman exponentiation) since these operators allow for (implicit) equality tests between arbitrary messages. In the transducer-based model, even one equality test (or alternatively, one application of the XOR operator) suffices for the undecidability.

Horn theories have also been used for the automatic analysis of *non-recursive* protocols (see, e.g., [5, 16] and references therein). The results and techniques employed in these works are very different to the ones presented here: The main goal of these works, which also consider operators with algebraic properties, is automatic protocol analysis w.r.t. an *unbounded* number of sessions, where, however, the intruder knowledge is over-approximated.

STRUCTURE OF THE PAPER. In the following section, we introduce our protocol and intruder model, with an example presented in Section 3. The undecidability and decidability results are stated in Section 4 and 5, respectively. We conclude in Section 6. We refer the reader to our technical report for full proofs [9].

2 The Protocol and Intruder Model

In this section, we introduce our protocol and intruder model, including messages, principals, protocols, and attacks, along the lines of [15] where, unlike the model in [15], here messages may contain the exclusive OR (XOR).

HORN THEORIES. Let Σ be a finite signature, V be a set of variables, and $\mathcal{T}$ denote the set of terms over Σ and V. *Ground terms* are terms without variables. Substitutions are defined as usual. The application of a substitution σ to a term t is denoted by $t\sigma$. Substitutions are defined on sets of terms and atoms (see below) in the obvious way. For a unary predicate q and a (ground) term $t \in \mathcal{T}$ we call $q(t)$ a *(ground) atom*. For a set S of terms we write $q(S)$ for the set $\{q(s) \mid s \in S\}$ of atoms. Let $\sim$ be a congruence relation over $\mathcal{T}$. We write $q(t) \sim q'(t')$ if $q = q'$ and $t \sim t'$. A *(unary) Horn theory* T is a finite set of Horn clauses of the form $a_1, \ldots, a_n \Rightarrow a_0$ with atoms a_i for every i. Given a set of ground atoms A and a ground atom a, we say that a can be derived from A w.r.t. T (written $A \vdash_T a$) if there exists a *derivation* for a from A using T, i.e., there exists a sequence $b_1, \ldots, b_l$ of ground atoms such that $b_l \sim a$ and for every $i \in \{1, \ldots, l\}$ we either have $b_i \in A$ or there exists a substitution σ and a Horn clause $a_1, \ldots, a_n \Rightarrow a_0$ in T such that $a_0\sigma \sim b_i$ and for every $j \in \{1, \ldots, n\}$ there exists $k \in \{1, \ldots, i-1\}$ with $a_j\sigma \sim b_k$.

MESSAGES. Let $\mathcal{A}$ be a finite set of constants (also called *atomic messages*), such as principal names, nonces, and keys, and let $\mathcal{K}$ be a subset of $\mathcal{A}$ (the set of keys). We assume that $0, \mathsf{Sec} \in \mathcal{A}$ and that there is a bijection $\cdot^{-1}$ on $\mathcal{K}$ which maps every public (private) key k to its corresponding private (public) key k^{-1}. Let $\Sigma_{\mathcal{A}}$ (or simply Σ) be a finite signature consisting of all constants from $\mathcal{A}$, the unary function symbol $\mathsf{hash}(\cdot)$ (*hashing*), and the following binary function symbols: $\langle \cdot, \cdot \rangle$ (*pairing*), $\{\cdot\}$. (*symmetric encryption*), $\{\!|\cdot|\!\}$. (*public key encryption*), and $\oplus$ (*exclusive OR*).

The set of terms over Σ and V is defined by the following grammar:

$$\mathcal{T} ::= \mathcal{A} \mid V \mid \mathsf{hash}(\mathcal{T}) \mid \langle \mathcal{T}, \mathcal{T} \rangle \mid \{\mathcal{T}\}_{\mathcal{K}} \mid \{\!|\mathcal{T}|\!\}_{\mathcal{K}} \mid \mathcal{T} \oplus \mathcal{T}.$$

Note that we assume atomic keys, i.e., keys used to encrypt messages are required to be constants. We denote by $\mathit{Var}(t)$ the set of variables occurring in t.

Ground terms, i.e., terms without variables are called *messages*. To model the algebraic properties of XOR, we consider the congruence relation $\sim$ on $\mathcal{T}$ induced by the following equational theory:

$$\begin{aligned} x \oplus y &= y \oplus x \\ (x \oplus y) \oplus z &= x \oplus (y \oplus z) \\ x \oplus x &= 0 \\ x \oplus 0 &= x \end{aligned}$$

For example, we have that $a \oplus b \oplus \{\!|0|\!\}_k \oplus b \oplus \{\!|c \oplus c|\!\}_k \sim a$. (Due to the associativity of $\oplus$ we often omit brackets and simply write $a \oplus b \oplus c$ instead of $(a \oplus b) \oplus c$ or $a \oplus (b \oplus c)$.)

PRINCIPALS AND PROTOCOLS. A *protocol step* consists of a protocol rule and a send program. A *protocol rule* is of the form $t \to q(s)$ where $t, s \in \mathcal{T}$ and q is some unary predicate symbol. A *send program* Φ is a unary Horn theory where every Horn clause is of one of the following forms:

$$q'(t) \Rightarrow q''(x) \qquad \text{with } x \in \mathit{Var}(t), \tag{1}$$

$$q'''(s) \Rightarrow I(s') \qquad \text{with } \mathit{Var}(s') \subseteq \mathit{Var}(s), \tag{2}$$

where I is a distinguished unary predicate symbol, which will model the network, and hence, the intruder, $q', q'', q''' \neq I$ are arbitrary (not necessarily different) unary predicate symbols, t is a linear term (i.e., every variable in t occurs at most once) which does not contain the symbol $\oplus$, and the terms s and s' may be non-linear and may contain $\oplus$. Intuitively, clauses of the form (1), called *push clauses*, allow a principal to recursively traverse a term from top to bottom (e.g., process a list). Clauses of the form (2), called *send clauses*, are used by a principal to perform checks on messages (by matching them against s) and to output messages on the network, as will be clearer from the following definition:

For a ground atom $q(m)$, we define the set of *terms sent using* Φ by

$$[\![q(m)]\!]_{\Phi} = \{m' : q(m) \vdash_{\Phi} I(m')\}.$$

Now, the intuition behind a protocol step which consists of a protocol rule $t \to q(s)$ and a send program Φ is that a principal, after having received a term $t\theta$ for some ground substitution θ, sends all the terms from the set $[\![q(s\theta)]\!]_\Phi$ on the network, i.e., to the intruder, by running the send program Φ.

The decidability result in [15] in the Horn theory model without XOR works if t in (1) is flat, i.e., is of the form $t = f(x_1, \ldots, x_n)$ where the variables x_i are not required to be different. We could also allow such terms in (1). However, (complex) linear terms are better suited for modeling protocols. It is easy to see that linear terms in (1) can be turned into flat form by using auxiliary predicate symbols.

A *principal* Π is a finite edge-labeled tree where every edge is labeled by a protocol step. If the protocol rule of a protocol step is of the form $t \to q(s)$, we require that every variable occurring in s also occurs in t or the left-hand side of a protocol rule preceding $t \to q(s)$ in the tree Π. We also assume, w.l.o.g., that the set of predicates used in send programs of different protocol steps are pairwise disjoint, except that I may be used in all of the send programs. The intuition is that if a principal waits at a node of the tree and receives a message, then she can apply one of the protocol steps whose left-hand side (i.e., the left-hand side of the corresponding protocol rule) matches with the incoming message, and after having run the corresponding send program, moves to the next node.

For a principal Π we call a sequence π of protocol steps a *run of* Π if π is a sequence of protocol steps obtained when traversing Π from the root to some node of Π (not necessarily a leaf).

A *protocol* P is a tuple $(\Pi_1, \ldots, \Pi_l)$ of principals Π_i. We assume, w.l.o.g., that the set of variables of protocol rules of different principals are disjoint.

Attack. The intruder is the standard Dolev-Yao intruder extended by the ability to apply the XOR operator [6, 2]. Formally, the intruder is modeled by the following Horn theory $T_\oplus$ where $k \in \mathcal{K}$ and $x, y \in V$:

$$
\begin{array}{lll}
I(x), I(y) \Rightarrow I(\langle x, y\rangle) & I(x), I(k) \Rightarrow I(\{x\}_k) & I(\{x\}_k), I(k) \Rightarrow I(x) \\
I(\langle x, y\rangle) \Rightarrow I(x) & I(x), I(k) \Rightarrow I(\{\!|x|\!\}_k) & I(\{\!|x|\!\}_k), I(k^{-1}) \Rightarrow I(x) \\
I(\langle x, y\rangle) \Rightarrow I(y) & I(x) \Rightarrow I(\mathsf{hash}(x)) & I(x), I(y) \Rightarrow I(x \oplus y)
\end{array}
$$

Given a protocol $P = (\Pi_1, \ldots, \Pi_l)$, a *protocol execution scheme* of P is a sequence of protocol steps $\pi = \pi_1, \ldots, \pi_n$ such that each π_i can be assigned to one of the principals $\Pi_1, \ldots, \Pi_l$ and such that, for every i, the subsequence of elements of π assigned to Π_i is a run of Π_i, i.e., π is an interleaving of runs of the Π_i.

Now, an *attack* on P is a pair (π, θ) where $\pi = ((t_i \to q_i(s_i), \Phi_i))_{i=1}^n$ is a protocol execution scheme of P and θ is a ground substitution of the variables in $Var(\{t_1, s_1, \ldots, t_n, s_n\})$ such that

$$
I(0), I([\![q_1(s_1\theta)]\!]_\Phi), \ldots, I([\![q_{i-1}(s_{i-1}\theta)]\!]_\Phi) \vdash_{T_\oplus} I(t_i\theta), \quad \text{for } i = 1, \ldots, n \tag{3}
$$

$$
I(0), I([\![q_1(s_1\theta)]\!]_\Phi), \ldots, I([\![q_n(s_n\theta)]\!]_\Phi) \vdash_{T_\oplus} I(\mathsf{Sec}) \tag{4}
$$

where $\Phi = \bigcup_{i=1}^{n} \Phi_i$ (recall that different send programs use disjoint sets of predicates, except that they all may use I). Condition (3) says that in every step of the protocol execution the intruder is able to derive the message expected by the respective principal and (4) says that at the end he is able to derive the secret Sec. Note that, w.l.o.g., initially the intruder only knows the constant 0: One can define a designated principal that expects to receive 0 and in return outputs messages the intruder is allowed to know, e.g., public keys. A protocol is called *insecure* if there exists an attack on it. Let $\textsc{Attack}_{general} = \{P \mid P \text{ is an insecure protocol}\}$ denote the corresponding decision problem.

3 An Example Protocol

To illustrate our model, we present a formal description of the Recursive Authentication (RA) Protocol [1]. In what follows, we abbreviate messages of the form $\langle m_0, \ldots, \langle m_{n-1}, m_n \rangle \cdots \rangle$ by $m_0, \ldots, m_n$ and messages of the form $\langle m, \mathsf{hash}(\langle k, m \rangle) \rangle$, i.e., a message m along with a keyed hash on m, by $\mathtt{hash}_k(m)$.

The key distribution server S of the RA protocol shares a long-term (symmetric) key with every principal and performs only one (recursive) protocol step in a protocol run. In this protocol step, S receives an a priori unbounded sequence of requests of pairs of principals who want to obtain session keys for secure communication and then generates so-called certificates which contain the session keys. An example of the kind of message S receives is

$$\mathtt{hash}_{K_c}(C, S, N_c, \mathtt{hash}_{K_b}(B, C, N_b, \mathtt{hash}_{K_a}(A, B, N_a, -))) \tag{5}$$

where N_c, N_b, and N_a are nonces generated by C, B, and A, respectively, and K_c, K_b, and K_a are the long-term keys shared between the server S and the principals C, B, and A, respectively. Recall that, for instance, $\mathtt{hash}_{K_a}(A, B, N_a, -)$ stands for the message $\langle \langle A, \langle B, \langle N_a, - \rangle \rangle \rangle, \mathsf{hash}(\langle K_a, \langle A, \langle B, \langle N_a, - \rangle \rangle \rangle \rangle \rangle) \rangle$. Message (5) consists of three requests and indicates that C wants to share a session key with S, B with C, and A with B. The constant "$-$" marks the end of the sequence of requests. We emphasize that messages sent to S may contain an arbitrary number of requests—which must be processed by S recursively. Now, given message (5), S processes the requests starting from the outermost. First, S generates two certificates for C, namely, $\langle C, S, K_{cs} \oplus \mathtt{hash}_{K_c}(N_c), \{C, S, N_c\}_{K_{cs}} \rangle$ and $\langle C, B, K_{bc} \oplus \mathtt{hash}_{K_c}(N_c), \{C, B, N_c\}_{K_{bc}} \rangle$ (from these certificates C can easily deduce K_{cs} and K_{bc} and check whether the encrypted messages have the expected form). In the same way, certificates for B and A are generated, where A only obtains one certificate (containing the session key for communication with B).

Formally, the protocol step performed by S is as follows, where we assume that $P_0, \ldots, P_n$ are the principals that may participate in the RA protocol, with $P_n = S$, and every P_i, $i < n$, shares a long-term key K_i with S: The protocol rule of S is simply $x \to q(x)$ and the send program consists of the following Horn

clauses, where $j \leq n$ and $i, i' < n$:

$$\begin{aligned} q(\langle\langle x_1, \langle x_2, \langle x_3, x_4\rangle\rangle\rangle, x_5\rangle) &\Rightarrow q(x_4) \\ q(\mathtt{hash}_{K_i}(P_i, P_j, x, -)) &\Rightarrow I(M_{i,j}) \\ q(\mathtt{hash}_{K_i}(P_i, P_j, x, \mathtt{hash}_{K_{i'}}(P_{i'}, P_i, x_1, x_2))) &\Rightarrow I(M'_{i,i'}) \\ q(\mathtt{hash}_{K_i}(P_i, P_j, x, \mathtt{hash}_{K_{i'}}(P_{i'}, P_i, x_1, x_2))) &\Rightarrow I(M_{i,j}) \end{aligned}$$

where $M_{i,j} = \langle P_i, P_j, K_{ij} \oplus \mathtt{hash}_{K_i}(x), \{P_i, P_j, x\}_{K_{ij}}\rangle$ and $M'_{i,i'} = \langle P_i, P_{i'}, K_{i'i} \oplus \mathtt{hash}_{K_i}(x), \{P_i, P_{i'}, x\}_{K_{i'i}}\rangle$. The server would also check whether the first request is addressed to it. This can easily be captured by using another predicate; however, for simplicity of presentation this is not modeled here. The model of the principals $P_0, \ldots, P_{n-1}$ of the RA protocol is rather standard as they do not need to perform recursive computations. We therefore omit their formal specification here.

4 Undecidability of the General Case

We prove the following theorem (see [9] for details):

Theorem 1. *The problem* $\textsc{Attack}_{general}$ *is undecidable.*

The main intuition behind the proof is that, by combining recursive computations and XOR, it can be checked whether a sequence $C_0, \ldots, C_n$ of configurations corresponds to an (accepting) computation of a Turing machine. More precisely, let $M_i = C_i, \ldots, C_n$. The intruder can guess a sequence $C_0, \ldots, C_n$ and then a single principal performing just one protocol step checks whether i) C_0 corresponds to the initial configuration, ii) C_n corresponds to a final configuration, and iii) C_{i+1} is a successor configuration of C_i, for every $i < n$. If i) and ii) are satisfied, the principal outputs $\{M_0\}_k$ and $\{M_n\}_k \oplus \mathsf{Sec}$, respectively, to the intruder. If iii) is satisfied, for i, the principal outputs $\{M_i\}_k \oplus \{M_{i+1}\}_k$ to the intruder. Now, if all checks—i), ii), and iii) for every $i < n$—are successful, and hence, the sequence of configurations is a valid computation, the intruder can XOR all messages obtained from the principal. This yields Sec. If at least one check failed, at least one "link" in the XOR chain would be missing in order to obtain Sec.

A similar reduction as the one sketched above also works if XOR is replaced by symmetric encryption where keys may be arbitrary messages (complex keys). Hence, we also obtain the following theorem, where $\textsc{Attack}_{compkey}$ is the security problem in our model with complex keys (and without XOR).

Theorem 2. *The problem* $\textsc{Attack}_{compkey}$ *is undecidable.*

5 Decidability of $\oplus$-linear Protocols

In the proof of undecidability (Theorem 1) we used that a principal may conjoin two messages by XOR where both messages may depend on messages received

from the network. In $\oplus$-linear protocols, defined next, this is forbidden. In this section, we show that the existence of attacks can be decided for $\oplus$-linear protocols.

A protocol P is *$\oplus$-linear* if for each subterm of the form $t \oplus s$ occurring in P (both in protocol rules and in send programs), t or s is ground. For example, if the term $(x \oplus a) \oplus y$ with $a \in \mathcal{A}$, $x, y \in V$ occurs in P, then P is not $\oplus$-linear. The RA protocol (Section 3) is an example of an $\oplus$-linear protocol; see, e.g., [2] for another example. Let $\text{ATTACK}_{\oplus-linear} = \{P \mid P \text{ is an } \oplus\text{-linear, insecure protocol}\}$.

The main result of this paper is:

Theorem 3. *The problem* $\text{ATTACK}_{\oplus-linear}$ *is decidable.*

Before we provide a proof sketch of this theorem, we note that our result extends the decidability results presented in [6, 2] for non-recursive protocols to recursive protocols, in case the protocols are $\oplus$-linear and restricted to atomic keys.

The proof of Theorem 3 consists of two main steps: First, we prove certain properties of derivations in the Horn theory model with XOR (Section 5.1). Based on these properties we then reduce the security problem to the case without XOR, which by [15] is decidable. In the reduction we use the ability of principals to perform recursive computations in order to mimic operations involving XOR. The reduction is sketched in Section 5.2. An initial (minor) step, not further discussed in this extended abstract, is to turn a protocol into simple form. This is used to combine all derivations carried out in (3) and (4) into a single derivation from $I(0)$ to $I(\mathsf{Sec})$. So, we may consider an attack as a single derivation, called *attack derivation* (see [9] for details).

5.1 Good Derivations

In this section, we identify and analyze properties of attack derivations.

First, we need to introduce some notation and terminology. We call a term *standard* if its top-symbol is not $\oplus$; otherwise, it is called *non-standard*. For a protocol P, let $\mathcal{S}_P$ denote the set of all the ground subterms of terms occurring in P and let $\mathcal{C}_P$ be the set consisting of all terms of the form $t_1 \oplus \cdots \oplus t_n$ with $t_i \in \mathcal{S}_P$ modulo $\sim$. Elements of $\mathcal{C}_P$ are referred to by c and decorations thereof. In what follows, non-standard terms will be written as $c \oplus t_1 \oplus \cdots \oplus t_n$ where c stands for a (ground) term in $\mathcal{C}_P$ and $t_1, \ldots, t_n$ are standard terms not in $\mathcal{C}_P$. A term is *$\mathcal{C}_P$-long* (or just *long*, if $\mathcal{C}_P$ is clear from the context) if it is of the form $c \oplus t_1 \oplus \cdots \oplus t_n$, for $n > 1$. Otherwise it is called *$\mathcal{C}_P$-short* (or just *short*).

We now introduce what we call modest and normal derivations and prove properties about them.

A derivation is *modest* if it uses the rules depicted in Figure 1 instead of $I(x), I(y) \to I(x \oplus y)$ where $c, c_0, \ldots, c_n \in \mathcal{C}_P$ and $t, t', t_1, \ldots, t_n$ are standard ground terms not in $\mathcal{C}_P$; s and s' are arbitrary ground terms. We observe that in a modest derivation long terms may only be used to obtain an element of $\mathcal{C}_P$, by applying rule (6). In all other rules ((7)–(10)), only short terms may

$$I(s), I(s') \to I(c) \quad \text{for } s \oplus s' \sim c \tag{6}$$

$$I(c \oplus t), I(c) \to I(t) \tag{7}$$

$$I(c \oplus t), I(c') \to I(c \oplus c' \oplus t) \quad \text{for } c \not\sim 0 \text{ and } c \oplus c' \not\sim 0 \tag{8}$$

$$I(t), I(c) \to I(c \oplus t) \tag{9}$$

$$I(c_0), I(c_1 \oplus t_1), \ldots, I(c_n \oplus t_n) \to I(c \oplus t_1 \oplus \cdots \oplus t_n) \quad \text{for } n > 1, \text{ where the terms } t_1, \ldots, t_n \text{ are pairwise distinct and } c \sim c_0 \oplus \cdots \oplus c_n \tag{10}$$

Fig. 1. $\oplus$-Rules in Modest Derivations

be conjoined by XOR. However, (10), which allows to combine an unbounded number of short terms, may produce a long term.

The next lemma states that it is enough to consider modest derivations. Therefore, in the remainder of this section we will assume derivations to be modest.

Lemma 1. *If there exists an attack on P, then this attack can be proven by a modest derivation.*

We now introduce normal derivations. In a normal derivation, applications of certain classes of rules are grouped into segments and segments occur in a certain order. In this extended abstract, we will only define some aspects of normal derivations (see [9] for a full definition).

If $b_1, \ldots, b_l$ is a derivation, then $b_i, b_{i+1}, \ldots, b_j$ for $i \leq j$ is a subsequence of the derivation. A *segment* of a derivation is a maximal subsequence which does not contain any atom of the form $I(c)$, for some $c \in \mathcal{C}_P$, or any atom obtained by a protocol rule.

Rule (8) is called *variant rule*. A *variant segment* of a derivation is a maximal subsequence of a segment containing only atoms obtained by variant rules. Among others, a *normal* derivation satisfies the following conditions: (i) it does not contain two atoms a, a' such that $a \sim a'$, (ii) each segment contains at most one variant segment, and (iii) variant rules do not use as a premise an atom obtained from a variant rule. We can show the following:

Lemma 2. *If there exists an attack on P, then there exists a modest and normal derivation for this attack.*

Because the cardinality of the set $\mathcal{C}_P$ is exponentially bounded w.r.t. the size of P, the number of segments in an attack derivation is also exponentially bounded in the size of P. Now, as a result of Lemma 2 and the definition of normal derivations we obtain:

Lemma 3. *In a modest and normal attack derivation the number of variant segments is exponentially bounded in the size of the protocol.*

We now show that the number of long terms can be bounded in attack derivations. The key is the notion of a *profile* of a standard term. A profile α is defined w.r.t. an attack derivation δ and consists of an element c in $\mathcal{C}_P$ and a natural number k. Roughly speaking, two standard ground terms satisfy the same profile if they behave similarly w.r.t. c in the k-th segment of δ. In particular, if two terms have the same profile, each of them can be used instead of the other one when long terms are constructed by Rule (10). So, for a given derivation δ and a profile α, we will fix a term t^{δ}_{α} and use it whenever a term of profile α is used to build a long term.

If $c \oplus t_1 \oplus \cdots \oplus t_k$, for $k > 1$, is a long term, the positions where $t_1, \ldots, t_n$ occur are called *unimportant*. We can show the following lemma:

Lemma 4. *If there exists an attack on a protocol, then there exists a normal attack derivation δ for this protocol such that whenever terms t, t' of the same profile α occur in δ at unimportant positions, then $t = t'$.*

We call derivations of the form described in Lemma 4 *good*. Now, from the definition of profiles it immediately follows that the number of different profiles is (exponentially) bounded in the size of the protocol. Together with Lemma 4 we obtain:

Corollary 1. If a term $c \oplus t_1 \oplus \cdots \oplus t_n$ occurs in a good attack derivation, then n is bounded exponentially in the size of the protocol. Furthermore, in a good attack derivation for a protocol, the number of distinct terms of the form $c \oplus t_1 \oplus \cdots \oplus t_n$, for $n > 1$, is bounded by some (computable) number M in the size of the protocol.

5.2 Reduction to the $\oplus$-Free Case

We now show how the security problem can be reduced to the $\oplus$-free case, i.e., given a protocol P we construct a protocol P^+ which does not contain $\oplus$ such that there exists an attack on P (in the sense defined in Section 2) iff there exists an attack on P^+ in the $\oplus$-free setting. The main steps are i) to represent terms with $\oplus$ by $\oplus$-free terms and ii) to mimic intruder rules involving $\oplus$ in the $\oplus$-free setting.

For i)—representing terms—we use additional constants: a new constant e and, for each equivalence class $[c]_{\sim}$, $c \in \mathcal{C}_P$, a new constant denoted by $[c]$. Now, for a term t, we obtain its $\oplus$–free representation, denoted by $\ulcorner t \urcorner$, by recursively applying to each non-standard subterm of t the following transformation: a subterm of the form $c \oplus t$ (recall that, according to our convention, $c \in \mathcal{C}_P$ and t is a standard term not in $\mathcal{C}_P$) is transformed into $\{t\}_{[c]}$, and a subterm of the form $c \oplus t_1 \oplus \cdots \oplus t_n$, for $n > 1$, is transformed into $\{\{t_1, \{\ldots, \{\langle t_{n-1}, t_n\rangle\}_e \ldots\}_e\}_e\}_{[c]}$. We also substitute every $c \in \mathcal{C}_P$ by the constant $[c]$.

Now, we turn to intruder rules involving $\oplus$ and show how they can be mimicked in the $\oplus$-free setting. By the results of Section 5.1, we may assume that attack derivations are modest, normal, and good. In particular, by Lemma 1, it suffices to mimic rules (6) to (10):

– *Rule* (7) *and* (9): These rules can easily be mimicked by ordinary intruder rules (decryption and encryption). Consider, for instance, rule (7): In the original attack atoms $I(c \oplus t)$ and $I(c)$ are used to obtain $I(t)$. Now, $I(\ulcorner t \urcorner)$ can be derived from $I(\ulcorner c \oplus t \urcorner) = I(\{\ulcorner t \urcorner\}_{[c]})$ and $I(\ulcorner c \urcorner) = I([c])$ by the standard decryption rule.
– *Rule* (6): The result of this rule is of the form $I(c)$ with $c \in \mathcal{C}_P$. Because there is a bounded number, say L, of elements in $\mathcal{C}_P$, we can mimic this rule by adding L principals to P each with a single protocol step of the form $\langle \{x\}_{[c]}, \{x\}_{[c']} \rangle \to I([c \oplus c'])$.
– *Rule* (10): The result of this rule is a long term and we know, by Corollary 1, that the number of such terms is bounded by a constant M which only depends on the size of the protocol, so, again, we can handle this case by adding a bounded number of principals each with a single protocol step of the form

$$\langle [c_0], \{y_1\}_{[c_1]}, \ldots, \{y_n\}_{[c_n]} \rangle \to I(\{\{y_1, \{\ldots, \{y_{n-1}, y_n\}_e \ldots\}_e\}_e\}_{[c]}).$$

– *Rule* (8): By Lemma 3, we know that the number of variant segments (i.e., blocks of atoms obtained by the variant rule) is bounded by a number N depending only on the protocol size and that no element obtained by a variant rule is necessary as a premise of a variant rule in the same variant segment. Hence, each of these variant segments can be handled by a protocol step of the following form: (Note that it performs recursive computation.)

$$z \to p(z) \qquad \text{with the following send program:}$$
$$p(\langle x, y \rangle) \Rightarrow p(y)$$
$$p(\langle x, y \rangle) \Rightarrow p'(x)$$
$$p'(\langle \{x\}_{[c]}, [c'] \rangle) \Rightarrow I(\{x\}_{[c \oplus c']}) \qquad \text{for } c, c' \in \mathcal{C}_P$$

More details on the construction of P^+ can be found in [9]. We can show:

Lemma 5. *For an $\oplus$-linear protocol P there exists an attack on P (in the sense of Section 2) if and only if there exists an attack on P^+ in the $\oplus$-free setting.*

Since the security of P^+ is decidable [15] and P^+ can effectively be computed from P, Theorem 3 follows. A more careful analysis of the complexity of our construction reveals that the size of P^+ is double exponential in the size of P. As the secrecy of P^+ can be decide in NexpTime [15], we obtain a 3-NexpTime upper bound, which, however, we believe can be reduced to NexpTime by a more careful construction and a refinement of the proof in [15].

6 Conclusion

In this work, we have proved that security (w.r.t. a bounded number of sessions) is decidable for the class of $\oplus$-linear protocols. This is the first decidability result

for recursive protocols involving algebraic properties of operators. We have also shown that relaxing certain assumptions of our model lead to undecidability of security. Our decidability result was obtained in a modular way by first reducing the problem of deciding security in the Horn theory model with XOR to the one without XOR and then using the existing decidability result for the latter model. We expect that the modular proof technique developed in this paper also helps to deal with other operators, such as Diffie-Hellman exponentiation.

References

1. J.A. Bull and D.J. Otway. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, Malvern, UK, 1997.
2. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *LICS 2003*, pages 261–270. IEEE, Computer Society Press, 2003.
3. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FSTTCS 2003*, volume 2914 of *LNCS*, pages 124–135. Springer, 2003.
4. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Commuting Public Key Encryption. *ENTCS*, 125(1):55–66, 2005.
5. H. Comon-Lundh and V. Cortier. New Decidability Results for Fragments of First-order Logic and Application to Cryptographic Protocols. In *RTA 2003*, volume 2706 of *LNCS*, pages 148–164. Springer, 2003.
6. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS 2003*, pages 271–280. IEEE, Computer Society Press, 2003.
7. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.
8. R. Küsters. On the Decidability of Cryptographic Protocols with Open-ended Data Structures. *International Journal of Information Security*, 4(1–2):49–70, 2005.
9. R. Küsters and T. Truderung On the Automatic Analysis of Recursive Security Protocols with XOR. Technial Report, 2007. Available from http://people.inf.ethz.ch/kuestral/publications_html/KuestersTruderung-TR-STACS-2007.pdf.
10. R. Küsters and T. Wilke. Automata-based Analysis of Recursive Cryptographic Protocols. In *STACS 2004*, volume 2996 of *LNCS*, pages 382–393. Springer, 2004.
11. L.C. Paulson. Mechanized Proofs for a Recursive Authentication Protocol. In *CSFW-10*, pages 84–95. IEEE Computer Society Press, 1997.
12. O. Pereira and J.-J. Quisquater. A Security Analysis of the Cliques Protocols Suites. In *CSFW-14*, pages 73–81, IEEE Computer Society Press, 2001.
13. P.Y.A. Ryan and S.A. Schneider. An Attack on a Recursive Authentication Protocol. *Information Processing Letters*, 65(1):7–10, 1998.
14. V. Shmatikov. Decidable Analysis of Cryptographic Protocols with Products and Modular Exponentiation. In *(ESOP 2004)*, volume 2986 of *LNCS*, pages 355–369. Springer, 2004.
15. T. Truderung. Selecting theories and recursive protocols. In *CONCUR 2005*, volume 3653 of *LNCS*, pages 217–232. Springer, 2005.
16. K.N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *CADE 2005*, volume 3328 of *LNCS*, pages 337–352. Springer, 2005.