Regular Protocols and Attacks with Regular Knowledge

Tomasz Truderung*

Loria-Inria-Lorraine, France
Institute of Computer Science, Wrocław University, Poland

Abstract. We prove that, if the initial knowledge of the intruder is given by a deterministic bottom-up tree automaton, then the insecurity problem for cryptographic protocols with atomic keys for a bounded number of sessions is NP-complete. We prove also that if regural languages (given by tree automata) are used in protocol descriptions to restrict the form of messages, then the insecurity problem is NexpTime-complete. Furthermore, we define a class of cryptographic protocols, called *regular protocols*, such that the knowledge which the intruder can gain during an unlimited number of sessions of a protocol is a regular language.

1 Introduction

Formal verification of cryptographic protocols has been attracting much attention in the recent years (see [10, 4] for an overview). It has been very succesful in finding flaws in cryptographic protocols. Althout the general verification problem is undecidable [6, 1, 7], there are interesting and important decidable variants [5, 6, 13, 2]. One of them is the insecurity problem of protocols analyzed w.r.t. a bounded number of sessions, in presence of the so-called Dolev-Yao intruder, which is NP-complete [13]. In this case, one assumes that the initial knowledge of the intruder is a finite set of terms.

In this paper, we prove the decidability of security for bounded number of sessions, when the initial knowledge of the intruder is a regular language, with the assumption that keys used in protocols are atomic. We show that if the initial knowledge of the intruder is given by a deterministic bottom-up tree automaton, then the existence of an attack remains NP-complete.

A regular language which represents the initial knowledge of the intruder can be an approximation or an exact representation of the set of messages which could have been intercepted during an unbounded number of prior executions of some protocols. In fact, approximating the knowledge of the intruder by means of finite tree automata or similar formalisms has been studied by several authors (see e.g. [9, 11]). As a complementary result we define also a class of cryptographic protocols, called *regular protocols*, such that the exact knowledge which the intruder can gain during an unbounded number of sessions of a protocol is a

* Partially supported by the RNTL project PROUVE-03V360 and by IST-2001-39252 AVISPA.

regular language given by an alternating tree automaton of polynomial size w.r.t. the size of the protocol. The security problem for such protocols is DexpTime-complete. As an immediate consequence we obtain also an NexpTime algorithm for deciding protocols which consist of two phases: in the first one, only regular rules can be used, these rules can be, however, executed an unbounded number of times. Then, in the second phase, non-regular rules (i.e. rules of an arbitrary form as long as they have atomic keys) can be used a fixed number of times.

We extend also our decidability result to *protocols which regular constraints*, i.e. protocols which impose some *well-formness* constraints on messages that can be sent. In [13], a receive-send action of a principal is described by a rewrite rule $t \to s$ (where t, s are terms). The meaning of such a rule is that a principal after receiving any ground instance $t\theta$ of t (for any ground substitution θ), replies $s\theta$. It is impossible to model the behaviour of a principal who replies only if the term $t\theta$ has some form which cannot be expressed by patern matching (e.g. if $t\theta$ is list of encrypted messages). Protocols with regular constraints allow us to express required form of messages by constraints of the form $x \in L$, where L is a regular language (given by some tree automaton). Such constraints may express some *integrity* requirements. For instance, a checksum for a message m can be simulated by a term $f(m)$ (where f is a new function symbol), which can be adequate, if checksums are collision-free. This approach can be however inadequate, when *weak* checksums (in which, given a checksum of a message, it is possible to produce another message that evaluates to the same checksum) are considered. Modeling the set $\{\langle m, c\rangle \mid c \text{ is the checksum of } m\}$ by a regular language, and using regular constraints can give more precise results.

We show that the insecurity problem of protocols with regular constraints, and with the initial knowledge of the intruder given by finite tree automata is NexpTime-complete (it is NexpTime-hard, when the used automata are deterministic, and remains in NexpTime even for alternating tree automata).

In this paper, we make the following abstractions. We use the Dolev-Yao model of the intruder [5], which is a standard practice in formal verification of cryptographic protocols. We formulate protocols in the rule-based model used in [2, 3, 13]. In the case of regular protocols, where unbounded number of sessions is considered, this model cannot express fresh nonces which have to be replaced by constants (or some other terms). It implies that some false attacks can be found. It should be mentioned, that this approach is also quite standard, since verification of protocols with nonces is undecidable even in the restricted case, where the size of messages is bounded.

Related work. The security problem of protocols when the initial knowledge of the intruder is given by finite automata has not been considered so far. Similarly, there are no previous decidability results for protocols with regular constraints.

There are, however, many results related to regular protocols defined in Sect. 5. Regular protocols are a generalization of regular unary-predicate programs proposed in [8]. They are also closely related to a class of monadic Horn theories defined in [15]. Our regular protocols are more general than the class $\mathcal{H}1$ defined in [12]. In [1], the authors specify a class of protocols (without nonces, and satisfying so called *independence* condition) which is DEXPTIME-hard. Regular protocols are also more general than this class.

Structure of the paper. Sect. 2 contains some basic definitions. In Sect. 3, we prove that the insecurity problem for bounded number of sessions, when the initial intruder knowledge is given by a deterministic tree automaton is NP-complete. Sect. 4 contains complexity results for protocols with regular constraints. In Sect. 5, we define regular protocols, and prove their properties.

2 Preliminaries

Terms and term-DAGs. Let $T(\Sigma, V)$ denote the set of terms over the signature Σ and the set of variables V. If $V = \emptyset$, then we can write $T(\Sigma)$ instead of $T(\Sigma, V)$. A term is *ground*, if it does not contain variables. A (ground) *substitution* is a mapping from variables to (ground) terms, which, in a natural way, is extended to a mapping from term to terms.

For a given signature Σ, a *term*-DAG D is a labelled directed acyclic ordered graph such that, if a node v is labelled with a function symbol f of arity n, then it has n ordered immediate successors $v_1, \ldots, v_n$. In such a case we write $v =_D f(v_1, \ldots, v_n)$. For a term-DAG D, and a vertex $v =_D f(v_1, \ldots, v_n)$, we recursively define the *term* $t(v, D)$ *represented by* v *in* D by the equation $t(v, D) = f(t(v_1, D), \ldots, t(v_n, D))$.

Unary definite logic programs. Let Σ be a signature, V be a set of variables, and P be a set of predicate symbols (we assume here that all predicates are unary). If $p \in P$, and $t \in T(\Sigma, V)$, then $p(t)$ is an *atomic formula*. An atomic formula $p(t)$ is *ground*, if t is ground. A *unary definite logic program* is a finite set of *clauses* of the form $a_0 \leftarrow a_1, \ldots, a_n$, where $a_0, \ldots, a_n$ are atomic formulas.

We will use the following notation. Let P be a unary definite logic program, let A, B be sets of ground atomic formulas. We write $A \vdash_P B$, if there exists *a proof of* B *with respect to* P *assuming* A, i.e. a sequence $a_1, \ldots, a_n$ of atomic formulas such that each element of B occurs in $a_1, \ldots, a_n$, and, for each $i = 1, \ldots, n$, we have either (i) $a_i \in A$, or (ii) there exists a clause $b_0 \leftarrow b_1, \ldots, b_m$ in P, and a substitution θ such that $a_i = b_0\theta$, and each of $b_1\theta, \ldots, b_m\theta$ occurs in $a_1, \ldots, a_{i-1}$.

For a set of atomic formulas A, and an atomic formula a, we write $A \vdash_P a$ for $A \vdash_P \{a\}$. We write also $\vdash_P B$ for $\emptyset \vdash_P B$. It is easy to show that $A \vdash_P a$, if and only if a is in the least Herbrand Model of $P \cup A$.

Messages, protocols, and intruder. *Messages* are ground terms over the signature Σ consisting of constants (*atomic messages* such as principal names, nonces, keys), and the following binary function symbols: $\langle \cdot, \cdot \rangle$ (*pairing*) $\{\cdot\}_\cdot$ (*symmetric encryption*), and $\{\cdot\}^p_\cdot$ (*public key encryption*), with the restriction that keys used in public key encryption are constants, i.e. that a term of the form $\{t\}^p_s$ is valid only if s is a constant. We assume that there is a bijection $\cdot^{-1}$ on atomic messages which maps every public (private) key k to its corresponding

$$I(\langle x,y\rangle) \leftarrow I(x), I(y), \qquad I(x) \leftarrow I(\langle x,y\rangle), \qquad I(y) \leftarrow I(\langle x,y\rangle), \tag{1}$$

$$I(\{x\}_y) \leftarrow I(x), I(y), \qquad I(x) \leftarrow I(\{x\}_y), I(y), \tag{2}$$

$$I(\{x\}^p_k) \leftarrow I(x), I(k), \qquad I(x) \leftarrow I(\{x\}^p_k), I(k^{-1}) \qquad \text{(for each key } k\text{)} \tag{3}$$

Fig. 1. T_I — Intruder Rules.

private (public) key k^{-1}. We assume that Σ contains special constant *Sec* (a secret). We will sometimes omit $\langle\cdot,\cdot\rangle$, and write, for instance, $\{t,s\}_k$ instead of $\{\langle t,s\rangle\}_k$.

A *principal* Π is a sequence $(r_i \to s_i)_{i=1}^n$ of *rules*, where, for each $i = 1,\dots,n$, we have $r_i, s_i \in T(\Sigma, V)$, for a set of variables V, and every variable in s_i occurs in $r_1,\dots,r_i$. By $|\Pi|$ we denote the number of rules of Π (i.e. the length of the sequence Π). A rule $(r \to s)$ is intended to specify receive-send action of a principal who after receiving $r\theta$, for a ground substitution θ, replies $s\theta$. A *protocol* is a finite set of principals. This method of representing principals and protocols follows [13, 3, 2], where examples of modeling protocols in this framework can be found. The set of variables occurring in a protocol P will be denoted by $Var(P)$.

In the Dolev-Yao model [5], the intruder has the entire control over the network. He can intercept and memorize messages, generate new messages and send them to participants with a false identity. We express the ability of the intruder to generate (derive) new messages from a given set of messages by the program T_I in Figure 1, where the predicate symbol I is intended to describe the intruder knowledge. The rules (1) express his ability to construct new messages by pairing known messages, and by deconstructing them. The rules (2) and (3) express his ability to crypt and decrypt messages, when he has appropriate keys. For a set A of messages, let $I(A) = \{I(t) \mid t \in A\}$. We will say that the intruder can *derive a message t from messages A*, if $I(A) \vdash_{T_I} I(t)$.

Now we give a definition of an *attack for a bounded number of sessions*. In an attack, the intruder chooses some execution order of the rules of the given protocol and then produces input messages for these rules. These input messages have to be derived from the intruder's initial knowledge and the output messages obtained so far. The aim of the intruder is to derive the secret message *Sec*. Note that in this definition of an attack, only security (or more precisely *secrecy*) is the concern. We do not study here properties like for instance authentication or liveness. If some number of interleaving sessions of a protocol is to be analyzed, then these sessions have to be encoded into the protocol, which is the standard approach when protocols are analyzed w.r.t. a bounded number of sessions (see, for instance [13, 2]).

Formally, given a protocol $P = \{\Pi_1,\dots,\Pi_l\}$, a *protocol execution scheme* is a sequence of rules $\pi = \pi_1,\dots,\pi_n$ such that each element of π can be assigned to one of the participants $\Pi_1,\dots,\Pi_l$, and, for each participant Π_k $(k = 1,\dots,l)$, the subsequence of the elements of π assigned to Π_k is $\Pi_k^1,\dots,\Pi_k^m$, for some

$m \leq |\Pi|$, where Π_k^i is the i-th rule of Π_k.[1] An *attack* with an initial knowledge A_0 is a pair (π, σ), where π is a protocol execution scheme, and σ is a ground substitution such that, for all $i = 1, \ldots, n$, we have

$$I(A_0), I(s_1\sigma), \ldots, I(s_{i-1}\sigma) \vdash_{T_I} I(r_i\sigma), \text{ and} \tag{4}$$

$$I(A_0), I(s_1\sigma), \ldots, I(s_n\sigma) \vdash_{T_I} I(\mathsf{Sec}). \tag{5}$$

A protocol is *insecure*, if there exists an attack on it.

Finite tree automata. We will express finite tree automata by means of unary logic programs. We say that a logic program T with a set of accepting predicate symbols Q_F is an *alternating finite tree automaton*, if each rule of T has the form

$$p_0(f(x_1, \ldots, x_n)) \leftarrow p_1(y_1), \ldots, p_m(y_m) \tag{6}$$

where $x_1, \ldots, x_n$ are distinct variables, and for each $i = 1, \ldots, m$, the variable $y_i \in \{x_1, \ldots, x_n\}$. A program is a *nondeterministic finite tree automaton*, if each its rule has the form

$$p_0(f(x_1, \ldots, x_n)) \leftarrow p_1(x_1), \ldots, p_n(x_n). \tag{7}$$

where $x_1, \ldots, x_n$ are distinct variables. A program T is a *deterministic bottom-up finite tree automaton*, if each its rule has the form (7), and for each function symbol f and each sequence of predicate symbols $p_1, \ldots, p_n$, the program contains at most one clause of the form (7). It is easy to see that, in this case, for each term t, there exists at most one predicate symbol p such that $\vdash_T p(t)$.

Let T with Q_F be an automaton. A term t is *accepted by* (T, Q_F), if $\vdash_T q(t)$, for some $q \in Q_F$. The set of terms accepted by (T, Q_F) will be denoted by $L(T, Q_F)$.

3 Attacks with Regular Knowledge

In this section we consider the insecurity problem of protocols analyzed w.r.t. a bounded number of sessions, assuming that the initial knowledge of the intruder is a regular language given by a finite tree automaton. We assume that keys (both in symmetric and public key encryption) are atomic, which is the only assumption not made in [13], where only keys used in public key encryption were assumed to be atomic (in other respects, the result presented here subsumes the decidability result from [13]).

The rest of this section is devoted to prove that, when the initial knowledge of the intruder is given by a deterministic bottom-up tree automaton, then the insecurity problem is NP-complete. The proof proceeds in two steps. First, in

[1] More formally, a sequence $\pi_1, \ldots, \pi_n$ of rules is a protocol execution scheme, if there is a function $f : \{1, \ldots, n\} \to \{1, \ldots, l\}$ such that, for each $k = 1, \ldots, l$, assuming that integers $i_1 < \cdots < i_m$ are all the elements of $f^{-1}(k)$, we have $\pi_{i_j} = \Pi_k^j$, for each $j = 1, \ldots, m$.

$$I_i(f(x_1,\ldots,x_n)) \leftarrow q_1(x_1),\ldots,q_n(x_n) \tag{8}$$

whenever $q_0(f(x_1,\ldots,x_n)) \leftarrow q_1(x_1),\ldots,q_n(x_n)$ is a rule of T, and $q_0 \in Q_F$;

$$\begin{array}{rrlr}
I_i(\langle x,y\rangle) \leftarrow I_j(x), I_k(y) & & \text{if } i \geq j,k & (9)\\
I_i(x) \leftarrow I_j(\langle x,y\rangle) & I_i(y) \leftarrow I_j(\langle x,y\rangle) & \text{if } i \geq j & (10)\\
I_i(x) \leftarrow I_j(\{x\}_a) & I_i(x) \leftarrow I_j(\{x\}^p_{a^{-1}}) & \text{if } i \geq j, \text{ and } a \in E^s_i, & (11)\\
I_i(\{x\}_a) \leftarrow I_j(x) & I_i(\{x\}^p_a) \leftarrow I_j(x) & \text{if } i \geq j, \text{ and } a \in E^s_i. & (12)
\end{array}$$

Fig. 2. The Stage Theory for T and $\boldsymbol{e}$.

Section 3.1, we introduce *stage theories of protocols* which allow us to represent attacks in more uniform way. Next, in Section 3.2, we introduce the notion of ADAGs which are labelled term-DAGs suitable to represent attacks. We show that if an ADAG exists, then there exists an ADAG of polynomial size. It gives rise to the nondeterministic polynomial-time algorithm for the insecurity problem.

3.1 Stage Theories

In this subsection we express the existence of an attack, using *a stage theory of a protocol* which takes into account the fact that A_0 is a regular language represented by a logic program (and hence A_0 and the intruder inference rules can be represented in a uniform way). Second, instead of representing the knowledge of the intruder by the predicate I, the family of predicate symbols $I_0,\ldots,I_m$ is used to represent his knowledge at different stages of an attack.

Let P be a protocol, and A_0 be the initial knowledge of the intruder, represented by a finite tree automaton (T, Q_F). Let $\mathcal{K}$ be the set consisting of the constant Sec, and all the keys of the given protocol. We can assume without loss of generality that no rule of P have the form $a \to s$, for $a \in \mathcal{K}$ (if it is the case, we can replace it by e.g. $\langle a,a\rangle \to s$, obtaining a protocols which is equivalent w.r.t. the existence of an attack).

Let $\pi = (r_i \to s_i)^n_{i=1}$ be a protocol execution scheme, and $\Omega = \mathcal{K} \cup \{1,\ldots,n\}$. A sequence $\boldsymbol{e} = e_1,\ldots,e_m$ of elements of Ω is called a *stage sequence for* π, if $\boldsymbol{e}$ contains all the elements $\mathit{Sec}, 1,\ldots,n$, and whenever $e_i = k$ and $e_j = l$, for $i < j$, then $k < l$.

For $e \in \Omega$, let us define e^r, and e^s by the equations $e^r = r_e$, $e^s = s_e$, if $e \in \{1,\ldots,n\}$, and $e^r = e^s = e$, otherwise. Let $E^r_i = \{e^r_1,\ldots,e^r_i\}$, and $E^s_i = \{e^s_1,\ldots,e^s_i\}$. The set E^s_i represents keys and terms of the form $s_j\sigma$ available to the intruder at the i-th stage of an attack. The set E^r_i represents keys and terms of the form $r_j\sigma$ which should be known to the intruder before the i-th stage. Let $T_{\boldsymbol{e}}$ denote the program T extended with the stage theory for T and $\boldsymbol{e}$ (Figure 2), where $Q_I = \{I_0,\ldots,I_m\}$ are fresh predicate symbols. The predicate symbol I_k is intended to describe the intruder knowledge at the k-th stage of an attack with a substitution σ, where the terms from $\{t\sigma \mid t \in E^s_k\}$ are available to him.

Lemma 1. *Let π be a protocol execution scheme, and σ be a ground substitution. The pair (π, σ) is an attack iff there is a stage sequence $\boldsymbol{e}$ for π such that*

$$I_1(e_1^s\sigma), \ldots, I_m(e_m^s\sigma) \vdash_{T_e} I_0(e_1^r\sigma), \ldots, I_{m-1}(e_m^r\sigma). \tag{13}$$

Proof. First, suppose that (13) holds, for some π, $\boldsymbol{e}$, and σ, and that Δ is a proof of it. Without loss of generality, we can assume that $I_k(t)$ occurs in Δ before $I_l(s)$, if $k < l$. Let Δ_i denote the subsequence of Δ containing only facts of the form $I_i(t)$, and let $\Delta_{\leq i}$ be the concatenation of $\Delta_1, \ldots, \Delta_i$. Let $\Delta^*_{\leq i}$ be the sequence obtained from $\Delta_{\leq i}$ by substituting I_k by I. One can show, by induction on i, that $\Delta^*_{\leq i}$ is a proof w.r.t. T_I which uses only assumptions from $I(A_0) \cup \{I(s_j\sigma) : s_j \in E_i^s\}$ (i.e. $\Delta^*_{\leq i}$ is a proof of $I(A_0) \cup \{I(s_j\sigma) : s_j \in E_i^s\} \vdash_{T_I} \emptyset$). Now, let k be any integer from $\{1, \ldots, n\}$. There exists i such that $e_i = k$. By the definition of E_i^s, we have $e_i^s = s_k \notin E_{i-1}^s$. Moreover, if $s_l \in E_{i-1}^s$, then $l < k$. So, $\Delta^*_{\leq i-1}$ is a proof w.r.t. T_I which uses only assumption from $I(A_0), I(s_1\sigma), \ldots, I(s_{k-1}\sigma)$. By the definition of Δ, we have $I_{i-1}(r_k\sigma) \in \Delta_{\leq i-1}$ (because $r_k = e_i^r$), hence $\Delta^*_{\leq i-1}$ is a proof of $I(A_0), I(s_1\sigma), \ldots, I(s_{k-1}\sigma) \vdash I(r_k\sigma)$. Similarly, we show that (5) holds. So, we can conclude that (π, σ) is an attack.

Now, suppose that we have an attack (π, σ). Let Π_i be a proof of (4), for $i = 1, \ldots, n$, and let Π_{n+1} be a proof of (5). We split each Π_k (for $k = 1, \ldots, (n+1)$) into the maximal (w.r.t. its length) sequence $\Pi_k^1, \ldots, \Pi_k^{m_k}$ such that the last element of Π_k^i, for $1 \leq i < m_k$, is of the form $I(a)$ for $a \in \mathcal{K}$, and this occurrence of $I(a)$ is the only one in $\Pi_1, \ldots, \Pi_{k-1}, \Pi_k^1, \ldots, \Pi_k^i$. We want to re-index the obtained sequence of Π_k^i, so let $\hat{\Pi}_1, \ldots, \hat{\Pi}_N = \Pi_1^1, \ldots, \Pi_1^{m_1}, \ldots, \Pi_{n+1}^1, \ldots, \Pi_{n+1}^{m_{n+1}}$.

For $i = 1, \ldots, N$, let Δ_i be the sequence of facts obtained from $\hat{\Pi}_i$ by substituting each $I(t)$ by $I_{i-1}(t)$, and let e_i be equal to k, if $\hat{\Pi}_i = \Pi_k^{m_k}$, for some k, and, otherwise, let e_i be a, where $I(a)$ is the last element of $\hat{\Pi}_i$. Finally, let $S = \{t \in A_0 \mid I(t) \text{ occurs in } \Pi_1, \ldots, \Pi_{n+1}\}$, and let Δ_0 be a proof of $\vdash_{T_e} I_0(S)$. One can prove that the concatenation of $\Delta_0, \ldots, \Delta_N$ is a proof of (13).[2] □

A proof is *normal*, if for each term t, it contains at most one fact of the form $I_k(t)$ (for some k). The following fact is easy to prove.

Lemma 2. *It holds* (13) *iff there is a normal proof Δ of*

$$I_1(e_1^s\sigma), \ldots, I_m(e_m^s\sigma) \vdash_{T_e} I_{i_1}(e_1^r\sigma), \ldots, I_{i_m}(e_m^r\sigma), \tag{14}$$

where, for each $k = 1, \ldots, m$, we have $0 \leq i_k < k$.

3.2 DAG of the Attack

Suppose that we have a protocol P, a protocol execution scheme $\pi = (r_i \to s_i)_{i=1}^n$, and a stage sequence $\boldsymbol{e}$ for π. We denote by $\mathcal{T}(P)$ the set of subterms of $\{r_i, s_i\}_{i=1}^n \cup \mathcal{K}$. Suppose that the initial knowledge of the intruder is given by a deterministic bottom-up automaton (T, Q_F) with the set of predicate symbols Q, and the set of accepting predicate symbols Q_F. Let Z be the set of elements of the form ϵ, and $I_k^{\downarrow}$, $I_k^{\uparrow}$ (for $0 \leq k \leq |\boldsymbol{e}|$).

[2] We use here the assumption that no rule of P is of the form $a \to s$, for $a \in \mathcal{K}$.

Definition 1. A DAG *of the attack* (an ADAG for short) for $P, \boldsymbol{e}$ is a tuple $\langle D, \alpha, \delta_1, \delta_2 \rangle$ where D is a term-DAG over Σ with the set of vertices V, $\delta_1 : V \to Q$, $\delta_2 : V \to Z$, and α is a mapping from $\mathcal{T}(P)$ to V such that

(i) if $\alpha(f(t_1, \ldots, t_n)) = v$, then $v =_D f(v_1, \ldots, v_n)$, and $\alpha(t_i) = v_i$, for $i = 1, \ldots, n$,
(ii) if $v_0 =_D f(v_1, \ldots, v_n)$, and $\delta_1(v_i) = q_i$, for $i = 0, \ldots, n$, then T contains the rule $q_0(f(x_1, \ldots, x_n)) \leftarrow q_1(x_1), \ldots, q_n(x_n)$,
(iii) if $\delta_2(v) = I_i^{\uparrow}$, then we have either (a) $\delta_1(v) \in Q_F$, or (b) for each child v' of v, $\delta_2(v') = I_j^{\downarrow}$ or $\delta_2(v') = I_j^{\uparrow}$, for some $j \leq i$, and if $v =_D \{v'\}_a$ or $v =_D \{v'\}_a^{p}$ then $a \in E_i^{\boldsymbol{s}}$,
(iv) if $\delta_2(v) = I_i^{\downarrow}$, then either (a) $v = \alpha(s_k)$, for $s_k = e_i^{\boldsymbol{s}}$, or (b) for some parent v' of v, $\delta_2(v') = I_j^{\downarrow}$, for some $j \leq i$, and if $v' =_D \{v\}_a$ or $v' =_D \{v\}_{a^{-1}}^{p}$, then $a \in E_i^{\boldsymbol{s}}$,
(v) if $v = \alpha(e_i^{\boldsymbol{r}})$, then $\delta_2(v) = I_j^{\downarrow}$ or $\delta_2(v) = I_j^{\uparrow}$, for some $j < i$.

The following lemma links the existence of an attack and the existence of an ADAG for a given protocol and stage sequence.

Lemma 3. *Let P be a protocol. There exists an attack on P iff there exists a stage sequence $\boldsymbol{e}$ and an* ADAG *for $P, \boldsymbol{e}$.*

Proof. Suppose that there is an attack (π, σ). By Lemma 1 and Lemma 2, there is a sequence $\boldsymbol{e}$, and a normal proof Δ of (14). Let D be the DAG representing all the terms of the form $t\sigma$, where $t \in \mathcal{T}(P)$ (i.e. for each term s of the form $t\sigma$, D contains a vertex v representing s). For $t \in \mathcal{T}(P)$, let $\alpha(t)$ be the vertex v such that $t(v, D) = t\sigma$. For a vertex v of D, let $\delta_1(v)$ be (the only) state which T assigns to $t_v = t(v, D)$. Let $\delta_2(v) = \epsilon$, if Δ does not contain $I_j(t_v)$, for any j. If $I_j(t_v)$ occurs in Δ, then let $\delta_2(v)$ be $I_j^{\uparrow}$, if $I_j(t_v)$ is obtained using (9) or (12), and let $\delta_2(v)$ be $I_j^{\downarrow}$, otherwise (in this case either $t_v = s_k\sigma$, for $s_k = e_j^{\boldsymbol{s}}$, or $I_j(t_v)$ is obtained in Δ using (10) or (11)). One can show that $\langle D, \alpha, \delta_1, \delta_2 \rangle$ is an ADAG.

Now, suppose that $\langle D, \alpha, \delta_1, \delta_2 \rangle$ is an ADAG for $P, \boldsymbol{e}$. Let $\sigma(x) = t(\alpha(x), D)$. We produce the following sequence of facts: First, we put all the fact of the form $I_k(t)$, where $\delta_2(v) = I_k^{\downarrow}$ (for some k), and $t = t(v, D)$, in such a way that $q(t)$ is before $q'(t')$, if $t > t'$. Second, we put all the facts of the form $p(t)$, where $\delta_1(v) = p$, for $t = t(v, D)$, and all the facts of the form $I_k(t)$ (for some k), where $\delta_2(v) = I_k^{\uparrow}$, for $t = t(v, D)$, in such a way that $q(t)$ is before $q'(t')$, if $t < t'$. One can prove that this sequence is a normal proof of (14), which by Lemma 1 and Lemma 2, implies that there exists an attack. □

Lemma 3 is a crucial step of our construction, because it characterizes the existence of an attack by a structure which is defined by some local properties ((i)–(v) of Definition 1). As we will see, it allows us to minimize ADAGs, roughly speaking, by merging vertices which are indistinguishable from the point of view of this local properties.

Let D be an ADAG. We say that $v \in V$ is *free*, if $v \neq \alpha(t)$, for each $t \in \mathcal{T}(P)$. Let $\delta(v) = (\delta_1(v), \delta_2(v))$. A vertex v is said to be a *push vertex*, if $\delta_2(v) = I_k^{\downarrow}$, for some k; otherwise it is a *non-push vertex.* A vertex v is a *top* vertex, if $\delta_2(v) = I_i^{\downarrow}$ (and so it is a push vertex), and $v = \alpha(s_k)$, for $s_k = e_i^{\mathsf{s}}$ (and so we do not have to use its parents in order to ensure that (iv) of Definition 1 is met).

Now, we will show that if there exists an ADAG (for some $P, \boldsymbol{e}$) then there exists an ADAG of polynomial size. The proof proceeds in two steps. First, in Lemma 4, we minimize the number of non-push vertices. It is a simple step which resembles the proof of pumping lemma for regular (tree) languages. In the second step (Lemma 5), we show how to minimize the number of push vertices. To explain this step, it is convenient to think that Item (iv) of Definition 1 allows us to transfer labels of the form $I_k^{\downarrow}$ down the ADAG, so that it can be used by pop vertices (Item (iii)). Now, roughly speaking, if a number of pop vertices is polynomially bounded, then a polynomially bonded number of push vertices is sufficient to transfer the necessary information from top vertices to pop-vertices (which is expressed by the *pushing relation* in the proof of Lemma 5).

Lemma 4. *If there is an* ADAG D*, then there is an* ADAG D' *with the same number of push vertices, and with the set of non-push free vertices of the size at most* $c = m \cdot (2n + 1)$*, where* n *is the length of* $\boldsymbol{e}$*, and* m *is the number of predicate symbols of* T*.*

Proof. Let v, v' be free non-push vertices of D with $\delta(v) = \delta(v')$. We can assume that $v \not< v'$ (if it is not the case, we can switch them). Let us remove v and replace it by v' (i.e. whenever v was a child of u, we make v' a child of u instead). One can show that in this way we obtain an ADAG. We repeat this step until there are no two distinct free non-push vertices with the same value of δ. □

Lemma 5. *If there is an* ADAG D*, then there is an* ADAG D^* *of polynomial size w.r.t. the size of the given protocol, and the program* T*.*

Proof. Suppose that D is an ADAG Let D' be the ADAG obtained from D using Lemma 4. Let W be the set of all the push vertices of D' which either are not free, or are children of some non-push vertices. Note that $|W| \leq 2c + |P|$, where c is the constant from Lemma 4.

For each non-top vertex v with $\delta_2(v) = I_k^{\downarrow}$, we chose one of its parents $h(v)$ such that $\delta_2(h(v)) = I_{k'}^{\downarrow}$, for some $k' \leq k$ (so $h(v)$ can be used to verify the point (iv) of Definition 1). We will write $v' \mapsto_h v$, if $v' = h(v)$, and denote the transitive closure of $\mapsto_h$ by $\mapsto_h^*$. We will call $\mapsto_h$ a *pushing relation* of D. Note that $\mapsto_h$ defines a forest such that the roots of its trees are top vertices, and every push vertex is a node of this forest. Let us denote this forest by T_h. For a push vertex v, let $G(v)$ be the set $\{w \in W \mid v \mapsto_h^* w\}$ (note that if $v \in W$, then $v \in G(v)$).

Now, we perform the following changes in D'. Let us set δ_2 to ϵ in each free push vertex v such that $G(v) = \emptyset$. One can show that in this way we obtain an ADAG Next, suppose that v, v' are distinct free vertices such that

$\delta(v) = \delta(v') = (q, I_k^{\downarrow})$ with $G(v) = G(v') \neq \emptyset$. Note that $v \mapsto_h^* v'$ or $v' \mapsto_h^* v$. We assume the former case. Let us remove v and replace it by v', and put $h(v') = h(v)$. Let $\delta_2(u)$ be set to ϵ in each push vertex u such that $v \mapsto_h^* u$, and $v' \not\mapsto_h^* u$. One can prove that what we have obtained is an ADAG Note that no vertex from W has been removed, and moreover, for $v \in W$, the value of $\delta(v)$ has not been changed.

We repeat this step until there are no two distinct free push vertices v, v' with $\delta(v) = \delta(v')$ and $G(v) = G(v')$. Note that each time we modify the ADAG we modify also its pushing relation. Let D'' be the ADAG obtained in this way, and let $\mapsto_{h''}$ be its pushing relation. Because W is polynomial, $T_{h''}$ is polynomial as well: this forest has at most $|W|$ leafs (each leaf is an element of W), and each its path is not longer than $|W| \cdot c$ (note that c is the number of distinct values of δ, and $|W|$ is the maximal number of distinct values of the function G on each path). Each push vertex of D'' is in $T_{h''}$, so the number of push vertices in D'' is polynomial. Let us apply Lemma 4 to D'' obtaining D^*. The number of push vertices is unchanged, and the number of free non-push vertices is polynomial. Thus D^* has polynomial size. $\square$

Theorem 1. *Protocol insecurity for a bounded number of sessions, with the initial knowledge of the intruder given by a deterministic bottom-up tree automaton is* NP*-complete.*

Proof. For deciding a protocol, we guess a protocol execution scheme, a sequence $\boldsymbol{e}$ for it, then we guess an ADAG of polynomial size (verifying whether such a guessed structure is an ADAG can be easily done in polynomial time). NP-hardness follows from NP-hardness of deciding protocols without composed keys, with the initial knowledge of the intruder given as a finite set [13]. $\square$

4 Protocols with Regular Constraints

Definition 2. A *protocol with regular constraints* is a tuple $(P, \mathcal{D})$, where P is a protocol, and $\mathcal{D}$ is a *domain assignment* which assigns a regular language $\mathcal{D}_x$ (the domain of x) to each variable $x \in \mathit{Var}(P)$.

For a protocol with regular constraint $(P, \mathcal{D})$, a pair (π, σ) is an *attack* on $(P, \mathcal{D})$, if it is an attack on P, and furthermore, for each $x \in \mathit{Var}(P)$, we have $x\sigma \in \mathcal{D}_x$.

We consider the problem of deciding protocols with regular constraints, where both the initial knowledge of the intruder, and languages $\mathcal{D}_x$ are given by finite tree automata. As we will see the choice of the type of automata (deterministic, nondeterministic, alternating) does not have any impact on the complexity of the problem: in all these cases the problem turns out to be NEXPTIME-complete.

Proposition 1. *The problem of deciding a protocol with constraints $(P, \mathcal{D})$, where the initial knowledge of the intruder and the languages $\mathcal{D}_x$, for $x \in \mathit{Var}(P)$, are given by alternating tree automata can be reduced to the problem of deciding a protocol (without constraints) with a regular initial knowledge of the intruder given by an alternating automaton.*

Proof. Suppose that $(P, \mathcal{D})$ is a protocol with regular constraints, and that $Var(V) = \{x_1, \ldots, x_m\}$. Let A_0 and $\{A_i\}_{i=1}^m$ be alternating tree automata which describe the initial knowledge of the intruder and the languages $\mathcal{D}_{x_i}$, respectively. We assume that these automata have disjoint sets of states, and that the accepting state of A_i is q_i (for $0 \leq i \leq m$). Let A denote the union of $A_0, \ldots, A_m$ with the accepting state q_0 (recall that it is the accepting state of A_0).

Let P' be the protocol P with one additional principal having the only rule

$$\mathsf{Sec}, \{x_1\}_{k_1}, \ldots, \{x_m\}_{k_m} \to \mathsf{Sec}',$$

where $k_1, \ldots, k_m$ and Sec' are fresh constants. Let A' be the automaton A with additional transitions that assign the state q_0 to a term $\{t\}_{k_i}$ only if t can be assigned the state q_i. One can show that the intruder with the initial knowledge given by A_0 can derive Sec in the protocol $(P, \mathcal{D})$, if and only if the intruder with the initial knowledge given by A' can derive Sec' in the protocol P'. $\square$

It is known that, for an alternating tree automaton, one can construct an equivalent deterministic bottom-up tree automaton of exponential size. Hence, Proposition 1, and Theorem 1 have the following consequence.

Theorem 2. *The insecurity of a protocol $(P, \mathcal{D})$ with the initial knowledge of the intruder and the languages $\mathcal{D}_x$ given by alternating tree automata is in* NEXPTIME.

One can show that the exponential bounded tiling problem (which is NEXPTIME-hard) can be reduced to the problem of deciding a protocol with regular constraints which use deterministic automata only. Thus we have the following result (the proof is given the extended version of this paper [14]).

Theorem 3. *The insecurity of a protocol $(P, \mathcal{D})$ with regular constraints is* NEXPTIME*-hard, even if the initial knowledge of the intruder and languages $\mathcal{D}_x$ are given by bottom-up deterministic tree automata.*

Let us note that the reduction given in the proof of Proposition 1 has the following property: if the initial knowledge of the intruder and the languages $\mathcal{D}_x$ are given by nondeterministic (but not alternating) tree automata, then the resulting automaton A' is also nondeterministic (does not use alternations). We can use this fact and Theorem 3 to obtain the following result, which shows that the assumption about the determinism of the automaton in Theorem 1 is essential.

Corollary 1. *The insecurity problem of protocols (without constraints) with the initial intruder knowledge given by nondeterministic tree automata is* NEXPTIME*-hard.*

5 Regular protocols

The aim of this section is to specify a (possibly general) class of protocols such that each protocol P in this class has the following property: the knowledge which the intruder can gain during an unbounded number of sessions of P is a regular language. The class defined here is closely related to regular unary-predicate programs defined in [8], and to a class of monadic Horn theories defined in [15].

In this section we consider the analysis w.r.t. unbounded number of sessions. We should note that in this case, the formalism used do describe protocols does not model nonces (in the case of a bounded number of sessions nonces can be modeled by constants). Hence, we can assume without loss of generality, that a protocol is just a set of (independent) rules[3], and that each of its rules $r \to s$ says that if the intruder knows a term $r\theta$, than it can also know $s\theta$, for any ground substitution θ.

Definition 3. A term s *covers* x *in a term* t, if either $s = x$, or $s = f(s_1, \ldots, s_n)$, for some $f \in \Sigma$, and each occurrence of x in t is in the context of one of $s_1, \ldots, s_n$.

For instance, $s = \langle \{x\}_b, y\rangle$ covers x in $t = \{\{x\}_b, \{y, \{x\}_b\}_a\}_a$ (because each occurrence of x in t is in the context of $\{x\}_b$), but s does not cover x in $\{\{x\}_c\}_b$. Note also that any term covers x in $\{y\}_a$.

Definition 4. Let φ be the function, which assigns a set of terms to a term, defined by the equations $\varphi(t) = \varphi(t_1) \cup \varphi(t_2)$, if $t = \langle t_1, t_2\rangle$, and $\varphi(t) = \{t\}$, otherwise.

For instance $\varphi(\langle \{b\}_k, \langle \{b, c\}_k, d\rangle\rangle) = \{\{b\}_k, \{b, c\}_k, d\}$.

Definition 5. A rule $r \to s$ is *regular*, if for each $s' \in \varphi(s)$ the following conditions hold: s' is linear, and each term $r' \in \varphi(r)$ can be assign a subterm $\gamma_{s'}(r')$ of s', such that:

(i) for each $r' \in \varphi(r)$ and each $x \in \mathit{Var}(s')$, the term $\gamma_{s'}(r')$ covers x in r',
(ii) for each $r', r'' \in \varphi(r)$, if a variable $y \notin \mathit{Var}(s')$ occurs in both r' and r'', then $\gamma_{s'}(r') = \gamma_{s'}(r'')$.

A *protocol is regular*, if it consists of regular rules only.

Example 1. The rule $r \to s$, where $r = \{N_A, x, B, \{x, A\}^p_{K_B}\}^p_{K_A}$ and $s = \{x, A\}^p_{K_B}$ is regular. In fact, for $\gamma_s(r) = x$, the conditions of Definition 5 hold (it is because x covers x in any term, and $\mathit{Var}(s) = \{x\}$; note also that $\varphi(r) = \{r\}$ and $\varphi(s) = \{s\}$). Similarly, one can easily check that each rule which has only one occurrence of a variable on the right-hand side, is regular.

[3] If it is not the case, each principal $\{r_i \to s_i\}_{i=1}^n$ can be transformed to n principals with rules $r_1, \ldots, r_i \to s_i$, for each $i = 1, \ldots, n$. It is easy to check that this transformation is correct in the following sense: the sets of messages the intruder can gain during an unbounded number of sessions of the original protocol and the protocol after the transformation are the same.

Example 2. The rule $\{\{x,y\}_a, z\}_b, \{z,z\}_c \to \{\{y,x\}_b, d\}_c, \{z, \{x,y\}_a\}_c$ is regular. To show it, let us denote the left hand side by r, and the right-hand side by s. Note that $\varphi(r) = \{r_1, r_2\}$, where $r_1 = \{\{x,y\}_a, z\}_b$ and $r_2 = \{z,z\}_c$, and $\varphi(s) = \{s_1, s_2\}$, where $s_1 = \{\{y,x\}_b, d\}_c$ and $s_2 = \{z, \{x,y\}_a\}_c$. Clearly, terms s_1 and s_2 are linear. So, let $\gamma_{s_1}(r_1) = \gamma_{s_1}(r_2) = \langle y,x\rangle$ (note that $\langle y,x\rangle$ is a subterm of s_1, because $\{y,x\}_b$ is a shorthand for $\{\langle y,x\rangle\}_b$), and $\gamma_{s_2}(r_1) = \gamma_{s_2}(r_2) = \langle z, \{x,y\}_a\rangle$. One can see that $\langle y,x\rangle$ covers x and y in r_1 and r_2. One can also see that $\langle z, \{x,y\}_a\rangle$ covers x, y, and z in r_1 and r_2.

Similarly, we can show that the rule $\{z, \{\{y\}_a, x\}_b\}_a \to \{\{x, \{y\}_a\}_b, z'\}_c$ is regular. The rule $\{\{x\}_b, y\}_a \to \{x, \{y\}_b\}_a$ is not regular.

Theorem 4. *The knowledge which the intruder can gain during an unbounded number of sessions of a regular protocol, can be described by an alternating tree automaton with the polynomial number of states w.r.t. the size of the protocol. Moreover, such an automaton can be computed in exponential time.*

Proof (sketch). First, we translate a given regular protocol to a logic program: for each rule $r \to s$, we produce clauses of the form $I(s') \leftarrow I(r_1), \ldots, I(r_n)$, where $s' \in \varphi(s)$, and $\{r_1, \ldots, r_n\} = \varphi(r)$. Suppose that T is a logic program obtained in this way Let $T' = T \cup T_I$. One can show that the knowledge that the intruder can gain during the protocol execution is the interpretation of I in the least Herbrand model of T'. Moreover one can show that each clause $s \leftarrow r_1 \ldots r_n$ of T' meets the following conditions: s is linear, and each term r_i (for $i = 1, \ldots, n$) can be assign a subterm $\gamma(r_i)$ of s, such that: (i) for each $i = 1, \ldots, n$, and each $x \in \mathit{Var}(s)$, the term $\gamma(r_i)$ covers x in r_i, (ii) for each $i, j = 1, \ldots, n$, if a variable $y \notin \mathit{Var}(s)$ occurs in both r_i and r_j, then $\gamma(r_i) = \gamma(r_j)$. We will call clauses of this form *regular*.

Now, T' can be translated to equivalent program T'' which consists of rules of the following form only:

$$p(f(x_1, \ldots, x_n)) \leftarrow p_1(t_1), \ldots, p_n(t_n), \quad \text{where } f(x_1, \ldots, x_n) \text{ is linear.} \tag{15}$$

In order to obtain T'', one can first eliminate clauses with the head of the form $p(x)$ (we assume that we have a fixed signature). Now, suppose that a clause has the form $p(\langle s_1, s_2\rangle) \leftarrow p_1(t_1), \ldots, p_n(t_n)$ (for other function symbols the proof proceeds similarly). Let γ be as in the definition of regular clauses. We divide the literals $p_1(t_1), \ldots, p_n(t_n)$ into three groups A, B, C such that $p(t_i) \in A$ iff $\gamma(t_i) = \langle s_1, s_2\rangle$, $t_i \in B$ iff $\gamma(t_i) \le s_1$, and $t_i \in C$ iff $\gamma(t_i) \le s_2$. We remove the rule, and add the following ones:

$$p(\langle x,y\rangle) \leftarrow A[s_1/x, s_2/y], p'(x), p''(y), \qquad p'(s_1) \leftarrow B, \qquad p''(s_2) \leftarrow C,$$

where p' and p'' are fresh predicate symbols. We recursively repeat this procedure for p' and p''. One can show that the size of T'' is polynomial w.r.t. the size of T.

Monadic Horn theories consisting of clauses of the form (15) are considered in [15], where it is shown that they can be finitely saturated by a sort resolution[4].

[4] One can also show that the program T'' is in the class $\mathcal{H}1$ defined in [12], and so, by Theorem 1 of [12], is normalizable.

We can proceed similarly. Roughly speaking, we saturate P', successively adding simpler clauses, and finally, we remove all the clauses which are not of the form (6) (see page 5). Thus the obtained program is just an alternating automaton. We show that the saturation process stops after at most exponential number of steps, and that the obtained program is equivalent to P. The detailed proof can be found in the extended version of the paper [14].

Theorem 5. *Secrecy of a regular protocol is* DEXPTIME*-complete.*

Proof. To decide a secrecy of a regular protocol, we build (in exponential time) an alternating tree automaton A of polynomial number of states which describes the knowledge of the intruder, and check whether $\mathsf{Sec} \in L(A)$, which can be done in exponential time.

We prove DEXPTIME-hardness by reduction of the emptiness of the intersection of regular tree languages given by n finite automata. We build a protocol that encode all these automata in such a way that the i-th automaton recognizes a term t iff the intruder knows the term $\{t\}_{k_i}$. We add the rule $\{x\}_{k_1}, \dots, \{x\}_{k_n} \to \mathsf{Sec}$ to the protocol. One can see that the protocol is insecure, iff the intersection of the given automata is not empty. ☐

By a very similar technique, regular protocols can be extended to work with regular constraints: we can encode a finite state automaton A by some regular rules so that $t \in L(A)$ iff $I(\{t\}_{k_A})$, and add terms of the form $\{x\}_{k_A}$ to the left-hand side of rules.

The results of this section and Sections 3 can be easily combined to achieve decidability of secrecy of the following two-phases protocols. Suppose that a protocol, which uses only atomic keys, consists of some regular rules P_1, and some rules P_2 of arbitrary form. The intruder can execute rules from P_1 unbounded number of times (building a knowledge which is a regular language), and then he can execute the rules of P_2 at most once. Because, for an alternating tree automaton, one can construct an equivalent deterministic bottom-up tree automaton of exponential size, by Theorems 4 and 1, the insecurity problem of such a protocol can be decided in NEXPTIME.

6 Conclusions

We have extended the decidability result for protocols analyzed w.r.t. a bounded number of sessions to the case when the initial knowledge of the intruder is a regular language. We have shown that if this language is given by a deterministic bottom-up automaton, then the insecurity problem of a protocol is NP-complete, assuming that complex keys are not allowed. We have showed also that if we add to protocols regular constraints which guarantee that messages have a required form, then the problem of deciding protocols is NEXPTIME-complete. These results can be a starting point for developing practical algorithms for detecting attacks with regular initial knowledge.

We have also defined a family of protocols such that the set of messages that the intruder can gain during unbounded number of sessions is exactly a regular language.

An open problem is decidability of the security of protocols with *complex keys* against attacks with regular initial knowledge.

References

1. R. M. Amadio and W. Charatonik, *On name generation and set-based analysis in the Dolev-Yao model.*, in CONCUR, vol. 2421 of Lecture Notes in Computer Science, Springer, 2002, pp. 499–514.
2. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani, *An NP decision procedure for protocol insecurity with XOR.*, in LICS, IEEE Computer Society, 2003, pp. 261–270.
3. Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron, *Extending the Dolev-Yao intruder for analyzing an unbounded number of sessions.*, in CSL, vol. 2803 of Lecture Notes in Computer Science, Springer, 2003, pp. 128–141.
4. H. Comon and V. Shmatikov, *Is it possible to decide whether a cryptographic protocol is secure or not?*, Journal of Telecommunications and Information Technology, special issue on cryptographic protocol verification, 4 (2002), pp. 5–15.
5. D. Dolev and A. Yao, *On the security of public-key protocols*, IEEE Transactions on Information Theory, 29 (1983), pp. 198–208.
6. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov, *Undecidability of bounded security protocols*, in Workshop on Formal Methods and Security Protocols (FMSP'99), 1999.
7. S. Even and O. Goldreich, *On the security of multi-party ping-pong protocols*, in Technical Report 285, Israel Institute of Technology, 1983.
8. T. W. Frühwirth, E. Y. Shapiro, M. Y. Vardi, and E. Yardeni, *Logic programs as types for logic programs*, in LICS, 1991, pp. 300–309.
9. T. Genet and F. Klay, *Rewriting for cryptographic protocol verification.*, in CADE, vol. 1831 of Lecture Notes in Computer Science, Springer, 2000, pp. 271–290.
10. C. Meadows, *Formal methods for cryptographic protocol analysis: Emerging issues and trends*, IEEE Journal on Selected Areas in Communication, 21 (2003), pp. 44–54.
11. D. Monniaux, *Abstracting cryptographic protocols with tree automata.*, in SAS, A. Cortesi and G. Filé, eds., vol. 1694 of Lecture Notes in Computer Science, Springer, 1999, pp. 149–163.
12. F. Nielson, H. R. Nielson, and H. Seidl, *Normalizable horn clauses, strongly recognizable relations, and spi.*, in SAS, vol. 2477 of Lecture Notes in Computer Science, Springer, 2002, pp. 20–35.
13. M. Rusinowitch and M. Turuani, *Protocol insecurity with a finite number of sessions, composed keys is NP-complete.*, Theor. Comput. Sci., 1-3 (2003), pp. 451–475.
14. T. Truderung, *Regular protocols and attacks with regular knowledge. Extended version*, 2005. Available at `http://www.ii.uni.wroc.pl/~tt/papers/`.
15. C. Weidenbach, *Towards an automatic analysis of security protocols in first-order logic.*, in CADE, vol. 1632 of Lecture Notes in Computer Science, Springer, 1999, pp. 314–328.