Selecting Theories and Nonce Generation for Recursive Protocols*

Klaas Ole Kürtz
Christian-Albrechts-Universität
Kiel, Germany
kuertz@ti.informatik.uni-kiel.de

Ralf Küsters
ETH Zurich
Zurich, Switzerland
ralf.kuesters@inf.ethz.ch

Thomas Wilke
Christian-Albrechts-Universität
Kiel, Germany
wilke@ti.informatik.uni-kiel.de

ABSTRACT

Truderung's selecting theory model is one of the few models of cryptographic protocols which allows to model iterative (recursive) computations of principals and, at the same time, an automatic analysis in the following sense: there exists a procedure that checks whether in all runs of a given protocol a certain message is not revealed to an intruder. A major drawback of Truderung's model is that it allows only a finite number of constants, that is, there is no mechanism by which a principal can generate an unbounded number of fresh tokens such as nonces or session keys. We extend Truderung's model by such a mechanism and show that the extended model still allows an automatic analysis. We also demonstrate that the extended model is more appropriate to model the Recursive Authentication Protocol.

It is important to know that after publication of Truderung's result it became clear that his proof only works in a restricted setting. We show that there is no hope to find a remedy to this by proving that the secrecy problem in the unrestricted setting is undecidable. In light of this, the aforementioned result about extending Truderung's model is to be read as follows. Adding anonymous constants in the restricted setting leads to a model of cryptographic protocols where insecurity is decidable.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Network Protocols—*Protocol verification*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Mechanical Verification*; D.2.4 [**Software**]: Software/Program Verification—*Formal methods*

General Terms

Security, Verification, Theory

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

FMSE'07, November 2, 2007, Fairfax, Virginia, USA.

Keywords

Security protocols, automatic analysis, decidability

1. INTRODUCTION

In many cryptographic protocols, such as group protocols, principals perform recursive or iterative computations in one protocol step. Following [10], we call such protocols *recursive*, in contrast to *non-recursive* protocols, where the computations performed by principals do not require recursion. An example of a recursive protocol is the Recursive Authentication (RA) Protocol proposed by Bull and Otway [1]. In this protocol, a key distribution server receives a list of (arbitrarily many) requests of pairs of principals who want to establish session keys among them. The server processes this list iteratively, generates the session keys (one for each request), and then distributes the session keys. Another example of a recursive protocol is the Internet Key Exchange protocol (IKE) [6], in which a principal needs to pick a security association (SA) from an a priori unbounded list of SAs, where an SA is a collection of algorithms and other information used for encryption and authentication. An attack on IKE, found by Zhou [18] and independently Ferguson and Schneier [5], shows that when modeling recursive protocols, recursive message structures and computations on them must be taken into account, since otherwise some attacks might not be found. In other words, as also pointed out by Meadows [11], modeling recursive computations is security relevant.

Recently, first progress has been made in the study of the automatic analysis of recursive protocols [8, 10, 16, 14], where a bounded number of sessions is assumed, but no bound is put on the size of messages (see also *Further Related Work* at the end of this section). One appealing approach to the automatic analysis of such protocols is the one by Truderung [16], which is based on what we call the *selecting theory model* here, for which Truderung shows that the secrecy problem is decidable in non-deterministic exponential time when protocols are analyzed w.r.t. a bounded number of sessions.

In the selecting theory model, recursive/iterative computations performed by principals are modeled by certain Horn theories, so-called selecting theories, hence the name of the model. These selecting theories allow to model principals, such as the server in the RA protocol, which iteratively process a list or some other recursive structure, check whether the entries in the list have the expected form, and produce output messages for every entry. However, this model does

not provide a means to faithfully capture the generation of fresh nonces and keys. In the RA protocol, for example, the server generates a new session key for each new request in the list of requests. Since the number of requests in such a list is unbounded, the number of new session keys needed is unbounded as well. This example shows that modeling the generation of fresh nonces and keys is important for recursive protocols even if these protocols are analyzed w.r.t. a bounded number of sessions since in one protocol step an unbounded number of nonces and keys may be generated. The main goal of the present work is therefore to extend the selecting theory model to incorporate the generation of fresh nonces and keys in such a way that a more faithful modeling of recursive protocols is possible, but without giving up decidability of the secrecy problem.

Contributions.

The contributions of the present work are as follows. First, to incorporate the generation of fresh nonces and keys in the selecting theory model, we add to the signature of selecting theories an *infinite* set of what we call *anonymous constants*. We also extend the finite state kept by principals, represented as unary predicates in the selecting theory model, by registers which can store anonymous constants. The purpose of these registers is for principals to be able to remember anonymous constants from one iterative step to the next. Note that since there is an infinite number of anonymous constants, registers model some kind of infinite memory. Most importantly, in every iterative/recursive step, principals may write fresh constants (keys and nonces) into registers and the content of these registers can be written into output messages. We call this model the *extended selecting theory model*. In Section 3, we use it to specify the RA protocol with a faithful formalization of fresh key generation.

The selecting theories that we use in our model are more restricted than those proposed by Truderung [16] in that terms on the left-hand side of Horn clauses have to be linear (every variable occurs at most once in a term) or flat (terms are of the form $f(x_1, ..., x_n)$ with the x_i's being variables, not necessarily distinct). In [16], arbitrary non-linear terms were allowed. Opposed to this, we show that in this case the secrecy problem is in fact undecidable, rather than in NEXPTIME, even without the extensions of the selecting theory model described above. Hence, the results proved in [16] only hold under the restrictions considered in the present work.

The main contribution of the present work is to show that the secrecy problem is decidable in non-deterministic double-exponential time in the size of the extended selecting theory model. The main structure of the proof is similar to the one in [16], but non-trivial extensions are necessary to take the generation of fresh anonymous constants into account.

Further Related Work.

In [8, 9, 10], transducers were used to model recursive computations of principals. The work in [10] even allows the generation of fresh keys and nonces. But the expressiveness of these transducer-based models is orthogonal to the selecting theory model: While the transducer-based models allow principals to output messages of complex structure, in the selecting theory model only lists (or sets) of messages of a more simple structure can be produced. Conversely, the main disadvantage of the transducer-based model is that, unlike the selecting theory model, messages cannot be tested for equality without losing decidability. Finally, the proofs of decidability in these models employ, unlike the proofs in the selecting theory model, automata-theoretic techniques and the best upper bounds known for the decision procedure in the transducer-based model are non-elementary.

In [14], decidability of the secrecy problem for an extension of the selecting theory model by algebraic properties of the exclusive OR (XOR) has been shown. The proof is by a reduction from the case with XOR to the case without XOR. A similar reduction might also work for the extension of the selecting theory model considered in the present paper. We leave such a reduction to future work.

Horn theories have also been used for the automatic analysis of *non-recursive* protocols (see, e. g., [3, 17] 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 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 Section 2, we introduce our protocol and intruder model, with an example provided in Section 3. The main results are summarized in Section 4. We give a sketch of the decidability proof in Section 5 and present the technical heart of the proof in 6. The undecidability proof is given in Section 7. We conclude in Section 8. We refer to [7] for full details.

2. PROTOCOL AND INTRUDER MODEL

In this section, we describe Truderung's model of recursive protocols, i.e., the selecting theory model, and our extension, which allows the generation of fresh nonces and keys.

2.1 Signatures, Terms, and Substitutions

In our model, as in many other Dolev-Yao based models, messages are terms. To model an infinite supply of nonces and keys, we extend the finite signature of terms with an infinite set of constants.

Let Σ be a finite signature with function and constant symbols. By Σ_i we denote the subset of all function symbols of arity i in Σ. Let Γ denote a countably infinite set of constant symbols, the set of *anonymous constants*, with $\Gamma \cap \Sigma = \emptyset$.

Let $X, Y, Y^\star$ be disjoint finite sets of variables: The elements of X are called *regular variables*, the elements of Y *anonymous variables*, and the elements of $Y^\star$ are called *fresh variables*. We require that only anonymous constants can be assigned to anonymous and fresh variables.

Let $V = X \cup Y \cup Y^\star$. For a given term t, let $\mathrm{var}(t)$ denote the set of all variables occurring in t. For each $V' \subseteq V$, let $\mathrm{var}_{V'}(t) = \mathrm{var}(t) \cap V'$.

The set of terms over a signature $\bar{\Sigma}$ and a set of variables $\bar{V}$ will be denoted by $T(\bar{\Sigma}, \bar{V})$; we will later need the sets $T_X = T(\Sigma \cup \Gamma, X)$ and $T_V = T(\Sigma \cup \Gamma, X \cup Y \cup Y^\star)$. The *depth* of a term $t \in T(\bar{\Sigma}, \bar{V})$ is recursively defined by

$$\mathrm{depth}(c) = 0 \quad \text{for } c \in \bar{\Sigma}_0 \cup \bar{V} \ ,$$

$$\mathrm{depth}(f(t_1, ..., t_m)) = 1 + \max\{\, \mathrm{depth}(t_j) \mid j \in \{1, ..., m\}\} \quad \text{for } t_1, ..., t_m \in T(\bar{\Sigma}, \bar{V}), f \in \bar{\Sigma}_m \ .$$

A term t with $\text{depth}(t) = 1$ is called *flat*. A term is called *linear* if every variable occurs at most once in this term. A term is *simple* if it is linear or flat. A term t is called ground if $\text{var}(t) = \emptyset$.

A *(ground) substitution* is a mapping $\sigma\colon X \to T_X$ from regular variables to (ground) terms. A substitution σ is extended to a function $\hat{\sigma}\colon T_X \to T_X$ from terms to terms as usual. Since no confusion can arise, we will refer to $\hat{\sigma}$ as a substitution and simply write σ instead of $\hat{\sigma}$.

A function $\sigma\colon V \to T_V$ is called *anonymous substitution* if $\sigma(y) \in \Gamma$ for all $y \in Y \cup Y^\star$ and $\sigma(y_1^\star) \neq \sigma(y_2^\star)$ for all $y_1^\star, y_2^\star \in Y^\star$ with $y_1^\star \neq y_2^\star$. An anonymous ground substitution is defined analogously, and both are extended to mappings on terms as above.

2.2 Registers

When processing a message, our principals will have access to a special memory, called *register sequence*, consisting of a finite number of *registers*. Each register can store an anonymous constant. The principals can read constants from that memory, move constants within the memory, and generate anonymous constants in any register. We will use tuples of variables and anonymous constants to model all this.

A tuple $\kappa \in \Gamma^Z$ for $Z \in \mathbb{N}_{>0}$ will be called a *ground register sequence* with Z registers, a tuple $\kappa \in (Y \cup Y^\star)^Z$ will be called a *register sequence of variables*. We will denote the ith element of κ by $\kappa[i]$. We will use $\text{var}(\kappa)$ to denote the set of variables in κ.

2.3 Horn Theories and Proofs

In the (extended) selecting theory model, receive-send actions are described by Horn theories of a specific form. These Horn theories model recursive computations of principals. Before we introduce (extended) selecting theories, we recall the definition of Horn theories and proofs, where, however, we already extend predicates by registers and a mechanism to generate new anonymous constants (nonces and keys).

Let P be a set of unary predicate symbols and R be a set of binary predicate symbols. If $p \in P$ and $r \in R$, we say that $p(t)$ and $r(t, \kappa)$ are *atomic (ground) formulas* if $t \in T_V$ is a (ground) term and κ a (ground) register sequence; atomic ground formulas are also called *(Horn) facts*. A *Horn theory* Φ is a finite set of *Horn clauses* of the form $[a_1, ..., a_m \to a_0]$ where $a_0, ..., a_m$ are atomic formulas and where fresh variables, i.e., variables from $Y^\star$, only appear in a_0.

Let Φ be a Horn theory, and let A and B be sets of facts. Let $\Pi = [b_1, ..., b_m]$ be a finite sequence of facts with $B \subseteq \{b_1, ..., b_m\}$. The sequence Π is called a *(Horn) proof of B with respect to Φ assuming A*, if the following holds for each $j \in \{1, ..., m\}$: Either b_j is an *assumed fact*, i. e., $b_j \in A$; or b_j is a *constructed fact*, i. e., there is a clause $[t_1, ..., t_n \to t_0^j] \in \Phi$ and an anonymous substitution σ_j such that there exists $\{t_1', ..., t_n'\} \subseteq \{b_1, ..., b_{j-1}\}$ with $\sigma_j(t_0^j) = b_j$ and $\sigma_j(t_i) = t_i'$ for all $i \in \{1, ..., n\}$ and such that the following condition is satisfied: For each $j \in \{1, ..., m\}$, let $C_j = \emptyset$ if b_j is an assumed fact, and let $C_j = \sigma_j(\text{var}_{Y^\star}(t_0^j))$ if b_j is a constructed fact. We require that these m sets are pairwise disjoint, and we will say that Π *assigns* the set $C = \bigcup_{j=1}^m C_j$. The intuition is that fresh variables on the right hand-side of a Horn clause are substituted by new anonymous constants.

If a proof Π as above exists, we write $A \vdash_\Phi^C B$. For a single atomic formula b we write $A \vdash_\Phi^C b$ instead of $A \vdash_\Phi^C \{b\}$. We will omit C and write $A \vdash_\Phi B$ if $A \vdash_\Phi^C B$ holds for some C.

2.4 Messages

Up to this point, all definitions have been rather general; from now on, we focus on a cryptographic setting. A signature Σ is called a *protocol signature* if it consists of a finite set K of constants, modeling keys, a constant $\dot{c}$ modeling the intruder's initial knowledge, a constant \$ modeling the secret, a binary function symbol $\langle \circ, \circ \rangle$ for pairing, two unary function symbols $\{\circ\}_k$ and $\{\!|\circ|\!\}_k$, for each key k, for symmetric and asymmetric encryption, respectively, two unary function symbols $\mathtt{h}(\circ)$ and $\mathtt{H}(\circ)$ for hashing, two unary function symbols $\mathtt{h}_k(\circ)$ and $\mathtt{H}_k(\circ)$, for each key k, for keyed hashing, and a finite number of additional constants depending on the protocol (e. g., for names of principals). Usually, we omit the inner pairing symbols and write $\langle m_1, ..., m_n \rangle$ instead of $\langle m_1, \langle m_2, ..., \langle m_{n-1}, m_n \rangle ... \rangle \rangle$. Hence, the pairing symbol is right-associative.

The term $\mathtt{h}(t)$ should be thought of as representing the hash value of t, while the term $\mathtt{H}(t)$ should be thought of as representing the message-hash pair consisting of t and $\mathtt{h}(t)$. This means, in particular, that we will make sure (in further definitions, see below) that it is possible to extract t from $\mathtt{H}(t)$, but not from $\mathtt{h}(t)$. The same applies to the keyed versions, i. e., $\mathtt{h}_k(t)$ models the MAC value of t under the key k, whereas $\mathtt{H}_k(t)$ models the message-MAC pair. Modeling message-hash pairs and message-MAC pairs in this way is useful in the context of protocol models where linearity is required in receive-send actions, see also [10].

Note that the keys are constants, that is, we work with atomic keys—as shown in [14], allowing complex keys leads to undecidability. We assume that there exists a bijection $\circ^{-1}$ on the set of keys which maps every public (private) key k to its corresponding private (public) key k^{-1}.

For a set of anonymous constants Γ, *messages* are ground terms over $\Sigma \cup \Gamma$.

For the rest of the paper, we fix a protocol signature Σ and an infinite set Γ of anonymous constants.

2.5 Selecting Theories

In most protocol models a principal is described by a sequence of receive-send actions, where each such action is a simple rewrite rule that expresses which message a principal *sends* to the network, that is, to the intruder, upon *receiving* a message. In Truderung's framework simple rewrite rules are generalized in order to model recursive receive-send actions, i. e., actions where upon receiving a message a recursive computation is performed in order to compute a set of output messages sent to the intruder. These recursive receive-send actions are modeled by certain Horn theories, the selecting theories. In what follows, these selecting theories are defined, along with our extension for modeling the generation of nonces and keys.

We will define three types of Horn clauses, whose purpose can informally be described as follows:

1. *Push clauses* allow principals to traverse messages in a recursive, top-down fashion. They can, for instance, be used to process lists elementwise. We will use register sequences to store or generate anonymous constants.

2. *Send clauses* allow principals to send messages to the network. A principal computes an output message using parts of the term it is currently processing and the current contents of the register sequence.

3. *Pop clauses* will provide a look-ahead when processing messages as they can simulate runs of a nondeterministic bottom-up tree automata when viewing messages as trees—similar to the look-ahead mechanism provided in [10]. This can be used, e. g., for simple type checking.

Let Q be a finite set of unary predicates, called *pop symbols*, and R be a set of binary predicate symbols, called *push symbols*. The sets R and Q are assumed to be disjoint. Let $\mathtt{I} \notin Q \cup R$ be a unary predicate symbol, which will model the network, and hence, the intruder's knowledge.

We define the following three types of Horn clauses—*pop clauses* (1 a), *push clauses* (1 b), and *send clauses* (1 c):

$$q_1(x_1), ..., q_m(x_m) \to q(f(x_1, ..., x_m)) \;, \tag{1 a}$$

$$q_1(t_r), ..., q_m(t_r), r(t_r, \kappa) \to r'(x_r, \kappa') \;, \tag{1 b}$$

$$q_1(t_\mathtt{I}), ..., q_m(t_\mathtt{I}), r(t_\mathtt{I}, \kappa) \to \mathtt{I}(s_\mathtt{I}) \;, \tag{1 c}$$

where κ, κ' are register sequences of variables with Z registers and with $\mathrm{var}(\kappa) \subseteq Y$ and $\mathrm{var}(\kappa') \subseteq \mathrm{var}(\kappa) \cup Y^\star$, $m \in \mathbb{N}$, pop symbols $q, q_1, ..., q_m \in Q$, push symbols $r, r' \in R$, function symbols $f \in \Sigma_m \cup Y$, variables $x_1, ..., x_m \in X$, simple terms $t_r, t_\mathtt{I} \in T(\Sigma, X)$ with $\mathrm{depth}(t_r) > 0$, a variable $x_r \in \mathrm{var}(t_r)$, and a term $s_\mathtt{I} \in T(\Sigma, \mathrm{var}(t_\mathtt{I}) \cup \mathrm{var}(\kappa))$.

A finite set Φ of such clauses is called *selecting theory over* (Q, R) *using anonymous constants with* Z *registers*.

2.6 Principals and Protocols

A principal will be defined to be a finite sequence of protocol steps and a protocol is basically a finite set of principals. The description of a protocol will also contain a selecting theory Φ with anonymous constants which is used to describe the recursive computation performed by principals in protocol steps.

Formally, a *protocol step* τ over (Q, R) as above is of the form (t, r, s), where $t, s \in T(\Sigma, X)$ are terms and $r \in R$ is a push symbol. We will often write $t \to r(s)$ instead of (t, r, s).

As mentioned above, the semantics of a protocol step $t \to r(s)$ is w.r.t. a selecting theory Φ over (Q, R) with anonymous constants associated with the protocol. To describe the semantics of $t \to r(s)$ assume that a message $\sigma(t)$ for a ground substitution σ is received. Now, the recursive computation performed by the protocol step starts in the state $r(\sigma(s), \kappa)$ where κ is a register sequence initialized with fresh anonymous constants, i. e., constants not used so far in the run of the protocol.

The output produced by the protocol step is a finite set of messages s' for which $r(\sigma(s), \kappa) \vdash_\Phi^C \mathtt{I}(s')$ holds, where C is a set of fresh anonymous constants not used so far and not used to construct any other output message s''. All output messages are sent on the network (i. e., to the intruder).

For example, in the RA protocol $t = s = x$ and $\sigma(x)$ is typically a sequence of requests. In this protocol, Φ describes how to process this sequence starting from $r(\sigma(x), \kappa)$ and which messages are output to the network (see Section 3 for more details).

Note that it is also possible to model non-recursive protocol steps of the form $t \to \mathtt{I}(s)$: For each such step τ, we take a new push symbol $r_\mathtt{I}^\tau \in R$ and add the send clause $r_\mathtt{I}^\tau(x, (y_1, ..., y_Z)) \to \mathtt{I}(x)$ to the selecting theory where $y_1, ..., y_Z$ are pairwise distinct anonymous variables. This allows us to express $t \to \mathtt{I}(s)$ by $t \to r_\mathtt{I}^\tau(s)$.

A *principal* $\vec{\tau}$ over (Q, R) is a finite sequence $[\tau_1, ..., \tau_N]$ of protocol steps such that for every $n \in \{1, ..., N\}$ and $\tau_n = (t_n, r_n, s_n)$ we have that $\mathrm{var}(s_n) \subseteq \bigcup_{i=1}^n \mathrm{var}(t_i)$.

A *protocol* over (Q, R) with Z registers is a pair $\mathcal{P} = (P, \Phi)$ consisting of a finite set of principals P and a selecting theory Φ using anonymous constants over (Q, R) with Z registers.

2.7 The Intruder, Attacks, and Security

Our intruder has the usual capabilities of a Dolev-Yao intruder [4], i. e., he controls the network and is able to (de)compose messages. The *intruder theory* $\Phi_\mathtt{I}$ is a Horn theory which contains the following Horn clauses to model the intruder, where $x, x_1, x_2 \in X$ and $k \in K$:

$$\mathtt{I}(\langle x_1, x_2\rangle) \to \mathtt{I}(x_1) \;, \qquad \mathtt{I}(\langle x_1, x_2\rangle) \to \mathtt{I}(x_2) \;, \tag{2 a}$$

$$\mathtt{I}(\{x\}_k), \mathtt{I}(k) \to \mathtt{I}(x) \;, \; \mathtt{I}(\{\!|x|\!\}_k), \mathtt{I}(k^{-1}) \to \mathtt{I}(x) \;, \tag{2 b}$$

$$\mathtt{I}(\mathtt{H}(x)) \to \mathtt{I}(x) \;, \qquad \mathtt{I}(\mathtt{H}(x)) \to \mathtt{I}(\mathtt{h}(x)) \;, \tag{2 c}$$

$$\mathtt{I}(\mathtt{H}_k(x)) \to \mathtt{I}(x) \;, \qquad \mathtt{I}(\mathtt{H}_k(x)) \to \mathtt{I}(\mathtt{h}_k(x)) \;, \tag{2 d}$$

$$\mathtt{I}(x), \mathtt{I}(\mathtt{h}_k(x)) \to \mathtt{I}(\mathtt{H}_k(x)) \;, \; \mathtt{I}(x_1), \mathtt{I}(x_2) \to \mathtt{I}(\langle x_1, x_2\rangle) \;, \tag{2 e}$$

$$\mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\{x\}_k) \;, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\{\!|x|\!\}_k) \;, \tag{2 f}$$

$$\mathtt{I}(x) \to \mathtt{I}(\mathtt{h}(x)) \;, \qquad \mathtt{I}(x) \to \mathtt{I}(\mathtt{H}(x)) \;, \tag{2 g}$$

$$\mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{h}_k(x)) \;, \qquad \mathtt{I}(x), \mathtt{I}(k) \to \mathtt{I}(\mathtt{H}_k(x)) \;. \tag{2 h}$$

For a set of messages A, let $\mathtt{I}(A) = \{\, \mathtt{I}(t) \mid t \in A\}$. We say that *the intruder can derive a message* t *from* A if $\mathtt{I}(A) \vdash_{\Phi_\mathtt{I}} \mathtt{I}(t)$.

To define attacks, we need the notion of an execution scheme of a protocol, which intuitively is an interleaving of protocol steps: Let $\mathcal{P} = (P, \Phi)$ be a protocol with M principals $\vec{\tau}^1, ..., \vec{\tau}^M$, where for each $m \in \{1, ..., M\}$ the principal $\vec{\tau}^m$ has N_m protocol steps $\tau_1^m, \tau_2^m, ..., \tau_{N_m}^m$. A sequence $\pi = [\pi_1, ..., \pi_N]$ is called an *execution scheme* for $\mathcal{P}$ if there is a function $f\colon \{1, ..., N\} \to \{1, ..., M\}$ that has the following property: $f^{-1}(m) = \{p_1, ..., p_{N'_m}\}$ with $p_1 < \cdots < p_{N'_m}$ and $N'_m \le N_m$ for all $m \in \{1, ..., M\}$, and $\pi_{p_n} = \tau_n^m$ for all $n \in \{1, ..., N'_m\}$.

Now, an attack on a protocol $\mathcal{P}$ is defined as follows. A pair (π, σ) consisting of a ground substitution σ and an execution scheme $\pi = [t_n \to r_n(s_n)]_{n=1}^N$ for $\mathcal{P}$ is called an *attack* on $\mathcal{P}$ if the following conditions hold:

$$r_n(\sigma(s_n), \kappa^{(n)}) \vdash_\Phi^{C_n} T_n \quad \text{for all } n \in \{1, ..., N\} \text{ and} \tag{3 a}$$

$$\{\mathtt{I}(\dot{\varsigma})\} \cup T_1 \cup \cdots \cup T_n \vdash_{\Phi_\mathtt{I}} \mathtt{I}(\sigma(t_{n+1})) \quad \text{for all } n \in \{0, ..., N\} \;, \tag{3 b}$$

where T_n, for $n \in \{1, ..., N\}$, is a finite set of facts of the form $I(m)$ (the set of messages sent on the network in step n of the attack), $t_{N+1} = \$$, and $C_0, \ldots, C_N$ are pairwise disjoint subsets of Γ with $C_0 = \kappa^{(1)} \cup ... \cup \kappa^{(N)}$, and $\kappa^{(n)}$ is a ground register sequence with $\kappa^{(n)} \cap \kappa^{(n')} = \emptyset$ for $n' \neq n$.

A protocol is called *insecure* if there is an attack on it, otherwise it is called *secure*. The *secrecy problem* is the problem of deciding whether a given protocol is insecure.

3. AN EXAMPLE

We illustrate our model by providing a specification of the Recursive Authentication Protocol (RA protocol) [1], which extends the two-party authentication protocol by Otway and Rees [13] to a key exchange protocol for an unbounded num-

ber of principals. We start with a brief description of the protocol.

Assume that there is a set of principals $\{P_1, ..., P_N\}$ of which P_n wants to communicate in a secure way with P_{n+1} for all $n \in \{1, ..., N-1\}$. The purpose of the RA protocol is to distribute symmetric session keys $K_{(n,n+1)}$ for secure communication between P_n and P_{n+1} for all $n \in \{1, ..., N-1\}$. To this end, it is assumed that there is a server P_{N+1} that shares a long-term key K_n with each of the principals.

The protocol assumes that the requests for session keys are chained: If a principal P_n receives a request m_{n-1}, he appends his request and sends the resulting message to P_{n+1}, more precisely, he sends the message m_n defined by $m_n = \mathtt{H}_{K_n}(\langle P_n, P_{n+1}, R_n, m_{n-1}\rangle)$ to P_{n+1}, where R_n is a nonce and $m_0 = \bot$ is a constant. Note that the message grows from principal to principal and that within one run of the protocol there is no limit on the length of the message.

When the server receives such a request chain, it generates all the necessary keys, and constructs a message

$$m'_n = \langle \{\langle K_{(n-1,n)}, P_{n-1}, P_n, R_n\rangle\}_{K_n}, \{\langle K_{(n,n+1)}, P_n, P_{n+1}, R_n\rangle\}_{K_n}\rangle$$

for each principal P_n, the so-called certificates. The server sends $m' = \langle m'_1, ..., m'_N\rangle$ to P_N, which extracts m'_N and forwards $\langle m'_1, ..., m'_{N-1}\rangle$ to P_{N-1}. This process is iterated until P_2 sends m'_1 to P_1. We note that since m' is sent over the network to P_N, i.e., m' is given to the intruder, and since the intruder can decompose m', from the point of view of the security of the protocol one can just as well consider m' to be the multiset $\{m'_1, ..., m'_N\}$.

To model the server, we only need one protocol step, namely, $x_S \to r_S(x_S)$, with the following selecting theory over $(\emptyset, \{r_S\})$ using anonymous constants with two registers where $n, m_1 \in \{1, ..., N\}$ and $m_2 \in \{1, ..., N+1\}$:

$$r_S(\mathtt{H}_{K_n}(\langle P_n, P_{m_2}, x_1, x_2\rangle), (y_1, y_2)) \to r_S(x_2, (y_2, y_\star)) \ , \tag{4 a}$$

$$r_S(\mathtt{H}_{K_n}(\langle P_n, P_{m_2}, x_1, \mathtt{h}_{K_{m_1}}(\langle P_{m_1}, P_n, x_2, x_3\rangle)\rangle), (y_1, y_2)) \to \mathtt{I}(\{\langle y_2, P_{m_1}, P_n, x_1\rangle\}_{K_n}) \ , \ \mathtt{I}(\{\langle y_1, P_n, P_{m_2}, x_2\rangle\}_{K_n}) \ , \tag{4 b}$$

$$r_S(\mathtt{H}_{K_n}(\langle P_n, P_{m_2}, x_1, \bot\rangle), (y_1, y_2)) \to \mathtt{I}(\{\langle y_1, P_n, P_{m_2}, x_1\rangle\}_{K_n}) \ . \tag{4 c}$$

The first Horn clause models that the server recursively processes the list of requests from the outermost to the innermost. When going from one request to the next, the anonymous constant in y_2 is kept (recall that each constant has to be sent to two principals). In addition, a new anonymous constant is generated and stored in $y_\star$. The second Horn clause is used to produce the certificates for P_n where the keys written into the certificates are those anonymous constants stored in the two registers. The last Horn clause deals with the last request, for which only one certificate needs to be generated. Recall that $\mathtt{H}_k(x)$ represents the pair consisting of x and the keyed hash value of x under the key k. Modeling this directly with $\langle x, \mathtt{h}_k(x)\rangle$ would lead to non-linear, non-flat terms, which in general make the secrecy problem undecidable (Theorem 2).

4. MAIN RESULTS

Recall that the *secrecy problem* is the problem of deciding whether a given protocol is insecure (see Section 2). We obtain the following theorem.

THEOREM 1. (DECIDABILITY OF THE SECRECY PROBLEM) *The secrecy problem for protocols using anonymous constants is decidable in nondeterministic double exponential time.*

The proof will be sketched in Section 5 below, and some details of it will be given in Section 6.

In our definition of the protocol model the terms that occur on left-hand sides of push and send clauses in selecting theories are required to be simple terms. The following theorem shows that this requirement is crucial, even if we do not allow anonymous constants, i. e., in the scenario of [16].

A protocol is said to be a *protocol without anonymous constants and hashing, but with arbitrary patterns* if (i) anonymous variables do not occur ($\Gamma = \emptyset$ and every register sequence is of size 0), (ii) the hash symbols do not occur ($\mathtt{h}, \mathtt{H}, \mathtt{h}_k, \mathtt{H}_k \notin \Sigma$), and (iii) the terms t_r and $t_\mathtt{I}$ on the left-hand sides of the definition of push and send clauses, respectively, are only required to be elements of $T(\Sigma, X)$ with $\mathrm{depth}(t_r) > 0$.

THEOREM 2. (UNDECIDABILITY OF THE SECRECY PROBLEM FOR ARBITRARY PATTERNS) *The secrecy problem is undecidable for protocols without anonymous constants and hashing, but with arbitrary patterns.*

The proof will be given in Section 7. This also shows that the result stated in [16] does not hold in its full generality.

5. PROOF SKETCH FOR THEOREM 1

Our proof of Theorem 1 is an extension and modification of Truderung's proof in [16] and consists of three steps (see [7] for more details):

1. As a first, technical step, we transform the characterization of an attack to a more compact one.
2. We give a characterization of attacks by directed graphs, that is, we show that a protocol is insecure if and only if there exists a directed graph satisfying certain conditions. Following [15], we call these graphs *ADAGs.*
3. We show that if, for a given protocol, there exists an ADAG at all (that is, if there is an attack), then there is also an ADAG whose size is double exponentially bounded in the size of the protocol.

This immediately leads to a non-deterministic double exponential-time decision procedure: guess a "small" graph and check whether it is an ADAG.

Step 1. For the first, technical step of the proof, we proceed as follows, mainly following the steps roughly outlined in [16]. First, we show that the selecting theory of a protocol and the intruder's theory can be merged into a new Horn theory, the so-called *theory of a protocol*, which has the following property: There exists an attack on a protocol if and only if for the theory of the protocol a condition similar to a combination of (3 a) and (3 b) holds. Second, we extend

the theory of a protocol to a selecting theory called *stage theory*, which does not model one protocol step at a time, but the whole run of a protocol in one application of the theory, and derive a new characterization of the insecurity of the protocol using the new stage theory.

Step 2. We then use the characterization by a stage theory to come up with an appropriate notion of ADAGs, which exactly characterizes the insecurity of a protocol: A protocol is insecure iff an ADAG exists.

An ADAG is a graph structure with special labelings satisfying a set of local properties only. Our ADAGs need special components that (i) guarantee the freshness of anonymous constants and (ii) store the values of all register sequences for each node and each predicate symbol. In addition, the local properties required of ADAGs have to be defined in such a way that they work well with register sequences.

Step 3. Recall that for the third step of the proof, we show a "small ADAG property": If there exists an ADAG for a protocol, then there exists a small (double exponentially bounded) ADAG. To this end, we first show that there is a certain kind of normal form for ADAGs. Following Truderung, we speak of *simple* ADAGs. Second, we prove that the number of nodes of a certain type, called *goals*, can be bounded, where, roughly speaking, goals are those nodes in the ADAG that correspond to messages sent to the intruder. Finally, we use cut-and-paste arguments to derive the desired double exponential bound. An additional difficulty compared to the proof of Truderung is that there is no a priori bound on the number of anonymous constants introduced during the run of a protocol (or in an attack). Hence, we have to establish a bound for the number of anonymous constants in order to obtain the small ADAG property. We note that while Truderung obtains a single exponential bound for the size of ADAGs, our bound is double exponential. This is due to the fact that our ADAGs have special components that take care of the register sequences for anonymous constants.

6. THE GRAPH OF AN ATTACK

As pointed out in the previous section, the technical core of our proof is an adapted definition of what Truderung calls a *directed acyclic graph of the attack* (ADAG) in [15, 16]. Our extended definition will be given in what follows.

The first step sketched in Section 5 is to compress the $2 \cdot N + 1$ conditions of (3 a) and (3 b) into one condition such that an attack can be described in only one Horn proof. To this end, we annotate the predicate symbols with what are called *stages*, and we extend the selecting theory of a protocol: A *stage theory* is obtained from a selecting theory by combining it with the intruder theory and annotating the involved clauses with stages.

We will give some semi-formal description of stage theories in Subsection 6.1, make some formal preparations in Subsections 6.2 to 6.5 and then define ADAGs in Subsection 6.6.

6.1 An Outline of Stage Theories

Let $\mathcal{P}$ be a protocol with an execution scheme π with N protocol steps, and let K be the set of keys in Σ. Let $E = \{0, ..., N\} \times \{0, ..., |K|\}$ be the *set of stages* of $\mathcal{P}$, let $<$ be the lexicographical ordering on the set of stages E, and let $\ll$ be the ordering on the first component:

$$(n, m) < (n', m') \text{ if } n < n' \text{ or } (n = n' \text{ and } m < m') , \quad (5\,\text{a})$$

$$(n, m) \ll (n', m') \text{ if } n < n' , \quad (5\,\text{b})$$

$$(n, m) \ll n' \qquad \text{if } n < n' . \quad (5\,\text{c})$$

A partial function $f \colon K \dashrightarrow E$ is called *stage mapping* for $\mathcal{P}$ if

$$f(k) \in \{ (n, m) \in E \mid n \neq 0, m \neq 0\} \quad (6)$$

for all k. Such a stage mapping witness during which step a key was derived by the intruder, i. e., given a key k with $f(k) = e$, this key was derived in some stage $e_k < e$ and therefore can be used from stage e onwards to derive other terms or keys.

For stage theories, we will use $Q \cup \widetilde{\mathtt{I}}$ with $\widetilde{\mathtt{I}} = \{ \mathtt{I}^e \mid e \in E\}$ as pop symbols and $\widetilde{R} = \{ r^e \mid r \in R \cup \{r_\mathtt{I}\}, e \in E\}$ as push symbols, where the definition of the intruder push symbol $r_\mathtt{I}$ is left out for clarity reasons.

The precise definition of the notion of a *stage theory* given in [7] is very complex; to unterstand ADAGs envision a stage theory Φ_f as a selecting theory over $(Q \cup \widetilde{\mathtt{I}}, \widetilde{R})$ where the clauses are variations of the ones in the selecting theory Φ of a protocol and the intruder's theory $\Phi_\mathtt{I}$, essentially parameterized by stages from a stage mapping f.

For example, corresponding to clauses (2 e) and (2 f) in the intruder theory, we add the following clauses to the stage theory for each stage $e \in E$, each key k with $f(k) \leq e$, and each $e', e'' \leq e$:

$$\mathtt{I}^{e'}(x), \mathtt{I}^{e''}(y) \to \mathtt{I}^e(\langle x, y \rangle) \quad (7\,\text{a})$$

$$\mathtt{I}^{e'}(x) \to \mathtt{I}^e(\{x\}_k) , \quad \mathtt{I}^{e'}(x) \to \mathtt{I}^e(\{\!|x|\!\}_k) . \quad (7\,\text{b})$$

6.2 Instances of Stage Theories

Let Φ_f and Ψ_f be stage theories. The theory Ψ_f is an *instance* of Φ_f, if each clause in Ψ_f is an instance of a clause in Φ_f, more precisely if for each clause $\psi \in \Psi_f$ there is a substitution $\sigma \colon X \to T(\Sigma, X \cup Y)$ and a clause $\varphi \in \Phi_f$ such that $\hat{\sigma}(\varphi) = \psi$ where $\hat{\sigma}$ is the extension of σ on clauses.

6.3 Term DAGs

Let $\bar{\Sigma}$ be a signature. Then $D = (V, F, \mu)$ is a *term DAG* over $\bar{\Sigma}$ if it consists of a finite set of nodes V, a set of edges $F \subset V \times V \times \mathbb{N}$ with an order in last component, a labeling function $\mu \colon V \to \bar{\Sigma}$, and the following holds: For a node $v \in V$ with $\mu(v) = f$ for a function symbol $f \in \bar{\Sigma}_m$, the node v has m ordered successors $v_1, ..., v_m$ in the DAG, i. e., $(v, v_j, j) \in F$ for all $i \in \{1, ..., m\}$.

In the above situation, we will write $v \doteq_D f(v_1, ..., v_m)$, and we will omit D if the DAG is clear from the context. In addition, we recursively define the notation

$$D(v) = c \qquad \text{if } v \doteq_D c , \quad (8\,\text{a})$$

$$D(v) = f(D(v_1), ..., D(v_m)) \qquad \text{if } v \doteq_D f(v_1, ..., v_m) . \quad (8\,\text{b})$$

A term DAG D is *minimized* if for all nodes v_1, v_2 in D, we have $D(v_1) \neq D(v_2)$.

6.4 Embeddings

Let $T \subseteq T(\Sigma \cup \Gamma, X \cup Y)$ be a set of terms. Let $D = (V, F, \mu)$ be a term DAG over $\Sigma \cup \Gamma$. A function $\alpha \colon \mathrm{sub}(T) \to V$ is a *D-embedding for T* if for all $v \in V$ and all $y \in$

$Y \cap \mathrm{sub}(T)$ we have

$$v \doteq \mathsf{c} \in \Gamma \qquad \text{if } v = \alpha(y)\ , \tag{9 a}$$

$$v \doteq f(v_1, ..., v_m) \text{ and } \alpha(t_j) = v_j \text{ for all } j \in \{1, ..., m\} \quad \text{if } v = \alpha(f(t_1, ..., t_m))\ . \tag{9 b}$$

Two embeddings α_1 and α_2 are *compatible* if we have $\alpha_1(x) = \alpha_2(x)$ for each regular and anonymous variable $x \in \mathrm{dom}(\alpha_1) \cap \mathrm{dom}(\alpha_2) \cap (X \cup Y)$.

For $v \in V$ and $t \in T$ we further write $t \mapsto v$ if there exists a unique embedding α for $\{t\}$ which is determined by $\alpha(t) = v$. We write $(t, t') \mapsto (v, v')$ if both $t \mapsto v$ and $t' \mapsto v'$ hold true and both embeddings are compatible.

6.5 Register Sequences

Let κ be a register sequence of variables and κ' be a ground register sequence. We denote the substitution which maps each variable $y \in \kappa$ to the corresponding anonymous constant $\mathsf{c} \in \kappa'$ by $\mathrm{subst}(\kappa, \kappa')$ if such a substitution is well-defined; i. e., if for each $y \in \kappa$ we have exactly one $\mathsf{c} \in \kappa'$:

$$\mathrm{subst}(\kappa, \kappa') = \big\{\ y \mapsto \mathsf{c} \ \big|\ \text{there is } z \in \{1, ..., Z\} \text{ such that } \kappa[z] = y \in Y \text{ and } \kappa'[z] = \mathsf{c} \in \Gamma\ \big\}\ . \tag{10}$$

Again, we will write $(\kappa_1, \kappa_2) \mapsto (\kappa_1', \kappa_2')$ if both substitutions $\sigma_1 = \mathrm{subst}(\kappa_1, \kappa_1')$ and $\sigma_2 = \mathrm{subst}(\kappa_2, \kappa_2')$ exists and are compatible, i. e., if for each $y \in \mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2)$ we have $\sigma_1(y) = \sigma_2(y)$.

For example, the application of a push clause $[\ldots r_1(t_1, \kappa_1) \to r_2(t_2, \kappa_2)]$ to a fact $r_1(t_1', \kappa_1')$ will result in a fact $r_2(t_2', \kappa_2')$ where we have $(\kappa_1, \kappa_2) \mapsto (\kappa_1', \kappa_2')$.

6.6 ADAGs

Informal Description.

The definition below is fairly technical, so we start with some intuition.

The overall goal of an ADAG is to represent an attack. An ADAG mainly consists of a term DAG representing all terms that occur in the run of a protocol. The nodes of the DAG are labeled with function symbols and constants, so for each node v there is a unique term t defined by the labeling of v and its descendants; we will say v *corresponds to* t. A function α embeds some crucial terms of the protocol, called static terms, to D.

In addition, the nodes are labeled with predicate symbols by a function δ: A node v corresponding to a term t is labeled with all predicate symbols p for which there is $p(t)$ or $p(t, \kappa)$ in the corresponding Horn proof. For each node and each symbol there may be an unbounded set of register sequences κ. In the ADAG they are represented by a function γ. Moreover, the freshness of a generated anonymous constant in a register sequence of a node is witnessed by a function λ through condition (iii c) below.

The set of conditions (i) to (iii) below guarantees the correspondence between ADAGs and Horn proofs of the attack condition for stage theories: Condition (i) ensures that the conclusion of the Horn proof is fulfilled by the ADAG, while condition (ii) guarantees a certain kind of normal form for Horn proofs. Furthermore, each fact in a Horn proof is either an assumed fact or constructed by a clause. Accordingly, each predicate symbol in the ADAG needs a justification. This is ensured by condition (iii): The fact is either an assumed fact (iii a), or there is a pop clause (iii b), a push clause (iii c), or a send clause (iii d) that justifies the fact. For the last two cases this justification is witnessed by a function β.

Formal Definition.

Let $\mathcal{P} = (P, \Phi)$ be a protocol, let $\pi = [t_n \to r_n(s_n)]_{n=1}^N$ be an execution scheme and f be a stage mapping for the set E of stages.

Let Q_f be the set of predicate symbols occurring in Φ_f and let $K_\mathcal{P}$ be the keys used in the protocol. By $T_\mathcal{P}$ we denote the set of *static terms* occurring in $\mathcal{P}$ defined by

$$T_\mathcal{P} = \{\mathrm{¢}, \$\} \cup \{\, t_n, s_n \mid n \in \{1, ..., N\}\} \cup K_\mathcal{P}\ . \tag{11}$$

Let $\mathcal{D}$ be the tuple $(D, \Psi_f, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ with

- a finite term DAG $D = (V, F, \mu)$ over $\Sigma \cup \widehat{\Gamma}$,
- a stage theory Ψ_f which is an instance of the stage theory Φ_f,
- a finite set of anonymous constants $\widehat{\Gamma}$,
- an embedding function $\alpha\colon \mathrm{sub}(T_\mathcal{P}) \to V$, which embeds $T_\mathcal{P}$ in D,
- a witness function $\beta\colon V \times Q_f \times \mathbb{N} \dashrightarrow V \times \mathbb{N} \times \Psi_f$,
- a register sequence function $\gamma\colon V \times Q_f \times \mathbb{N} \dashrightarrow \widehat{\Gamma}^Z$,
- a predicate symbol labeling function $\delta\colon V \dashrightarrow 2^{Q_f}$, and
- a freshness function $\lambda\colon \widehat{\Gamma} \dashrightarrow V \times Q_f \times \mathbb{N} \times \{1, ..., Z\}$.

Then $\mathcal{D}$ is called *DAG of the attack (ADAG)* for $\mathcal{P}$ and (π, f) if the following three conditions hold (see the explanation below for additional notation we use):

(i) For all $n \in \{1, ..., N\}$ and all keys $k \in \mathrm{dom}(f) \cap K$ we have

$$\mathtt{I}^e \in \delta(\alpha(t_n)) \text{ for a stage } e \ll n\ , \tag{i a}$$

$$\mathtt{I}^e \in \delta(\alpha(k)) \text{ for a stage } e < f(k)\ , \text{ and} \tag{i b}$$

$$\mathtt{I}^e \in \delta(\alpha(\$)) \text{ for a stage } e\ . \tag{i c}$$

(ii) For each node v we have

$$\mathtt{I}^e \in \delta(v) \text{ for at most one } e \in E\ .$$

(iii) For each node v and each predicate symbol $p \in \delta(v)$ at least *one* of the following conditions holds, i. e., p is justified because of one of the following four cases (iii a) to (iii d):

(iii a) the application of a protocol step (or the intruder's initial knowledge)

$$v = \alpha(\mathrm{¢}) \text{ with } p = \mathtt{I}^{(0,0)}$$

$$\text{or } v = \alpha(s_n) \text{ with } p = r_n^{(n,0)} \text{ for some } n \in \{1, ..., N\}\ ,$$

$$\gamma^\star(v, p) = \{0\},$$

$$\gamma(v, p, 0)[z] \neq \gamma(v, p, 0)[z'] \text{ for } z \neq z',\ \text{and}$$

$$\lambda(\gamma(v, p, 0)[z]) = (v, p, 0, z) \text{ for all } z \in \{1, ..., Z\}\ ;$$

(iii b) the application of an (intruder) pop clause with $p \in Q \cup \widetilde{\mathtt{I}}$

$$v \doteq f(v_1, ..., v_m) \text{ and there is a clause}$$

$$\psi = \big[p_1(x_1), ..., p_m(x_m) \to p(f(x_1, ..., x_m))\big] \in \Psi_f$$

$$\text{with } p_j \in \delta(v_j) \text{ for all } j \in \{1, ..., m\} \text{ and}$$

$$v_{j_1} = v_{j_2} \text{ for all } j_1, j_2 \in \{1, ..., m\} \text{ with } x_{j_1} = x_{j_2}\ ;$$

(iii c) the application of an (intruder) push clause with $p \in \widetilde{R}$ witnessed by β

$$\begin{aligned}
&\gamma^\star(v,p) \neq \emptyset, \text{ and for all } i \in \gamma^\star(v,p)\\
&\text{we have } v' \doteq f(v_1,...,v_m)\\
&\quad \text{with } v = v_j \text{ for some } j \in \{1,...,m\} \text{ such that}\\
&\beta(v,p,i) = (v', p_m, i', \psi),\\
&\quad \text{with } \psi = \big[p_1(t'),...,p_m(t',\kappa') \to p(x_j,\kappa)\big] \in \Psi_f,\\
&\qquad t' = f(x_1,...,x_m),\\
&\qquad p_j \in \delta(v') \text{ for all } j \in \{1,...,m\},\\
&\qquad v_{j_1} = v_{j_2} \text{ for all } j_1, j_2 \in \{1,...,m\}\\
&\qquad\quad \text{with } x_{j_1} = x_{j_2}, \text{ and}\\
&\qquad (\kappa,\kappa') \mapsto (\gamma(v,p,i), \gamma(v',p_m,i')), \text{ as well as}\\
&\lambda(\gamma(v,p,i)[z]) = (v,p,i,z') \text{ for some } z' \text{ with } \kappa[z] = \kappa[z']\\
&\qquad\quad \text{for all } z \text{ with } \kappa[z] \in Y^\star\ ;
\end{aligned}$$

(iii d) or the application of a send clause with $p \in \widetilde{\mathtt{I}}$ witnessed by β

$$\begin{aligned}
&\beta(v,p,0) = (v', p_m, i', \psi)\\
&\quad \text{with } \psi = \big[p_1(t'),...,p_m(t',\kappa') \to p(t)\big] \in \Psi_f, \text{ and}\\
&\qquad p_j \in \delta(v') \text{ for all } j \in \{1,...,m\}, \text{ as well as}\\
&(\sigma_\kappa(t), t') \mapsto (v, v') \text{ for } \sigma_\kappa = \mathrm{subst}(\kappa', \gamma(v', p_m, i'))\ .
\end{aligned}$$

For a node v and $p \in \delta(v) \cap \widetilde{R}$ we define

$$\gamma^\star(v,p) = \{\, i \in \mathbb{N} \mid (v,p,i) \in \mathrm{dom}(\gamma)\}\ . \tag{13}$$

By abuse of notation, we used β as a function $\beta\colon V \times Q_f \times \mathbb{N} \dashrightarrow V \times Q_f \times \mathbb{N} \times \Psi_f$, i. e., mapping a node v, a symbol p and a register sequence i not only to (v', i', φ), but to (v', p', i', φ) if p' is in φ, i. e., if we have $\varphi = [..., p'(t', \kappa') \to p(t)]$ or $\varphi = [..., p'(t', \kappa') \to p(t, \kappa)]$.

We can now express the following lemma, linking the existence of an ADAG to the insecurity of a protocol:

Lemma 1. (Characterization of Attacks by ADAGs) *Let $\mathcal{P} = (P, \Phi)$ be a protocol.*

1. *If (π, σ) is an attack on the protocol then there exists a stage mapping f and an ADAG $(D, \Phi, \widehat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$ for $\mathcal{P}$ and (π, f).*
2. *If there is an ADAG for $\mathcal{P}$ for an execution scheme and a stage mapping (π, f) then there exists an attack (π, σ) on the protocol for a substitution σ.*

7. PROOF SKETCH FOR THEOREM 2

The proof is by reduction from Post's correspondence problem (PCP). The reduction is split into two parts: we first reduce PCP to the secrecy problem for protocols over an extended signature and then reduce the latter to the secrecy problem we are interested in. The first reduction is the crucial one; it draws heavily on ideas described in [2, pages 118–120], which apparently first appeared in [12].

Step 1. Reduction of Post's correspondence problem to the secrecy problem for protocols over an extended signature. Let n be such that Post's correspondence problem is undecidable over an alphabet with n elements, and let $A = \{a_0, \ldots, a_{n-1}\}$ be such an n-element alphabet. To our signature Σ, we add a unary symbol a_i^1 for every $i < n$ and a ternary function symbol f. The resulting signature is denoted Σ'.

In addition, we add the following to the intruder theory:

$$I(x_1), I(x_2), I(x_3) \to I(f(x_1,x_2,x_3))\ , \tag{14 a}$$

$$I(x) \to I(a_i(x))\ , \qquad \text{for } i < n. \tag{14 b}$$

That is, the intruder can apply the additional symbols arbitrarily.

For every word $u = a_{i_0} \ldots a_{i_{l-1}}$ over A and variable x, we define the term $\bar{u}(x)$ by

$$\bar{u}(x) = a^1_{i_{l-1}}(a^1_{i_{l-2}}(\ldots(a^1_{i_1}(a^1_{i_0}(x))))\ , \tag{15}$$

which simply encodes the word as a term with a free variable.

Let $(u_0, v_0), \ldots, (u_{m-1}, v_{m-1})$ be a PCP instance. For every sequence $\pi = i_0, \ldots, i_l$ of indices $< n$, which we call index sequences for short, and every $j \le l$, we define inductively:

$$u_\pi^0 = u_{i_0}\ , \qquad u_\pi^{j+1} = u_\pi^j u_{i_{j+1}}\ , \tag{16 a}$$

$$v_\pi^0 = v_{i_0}\ , \qquad v_\pi^{j+1} = v_\pi^j v_{i_{j+1}}\ . \tag{16 b}$$

In addition, we let $u_\pi = u_\pi^l$ and $v_\pi = v_\pi^l$. In other words:
(i) An index sequence π is a solution to the given instance of the PCP iff $u_\pi = v_\pi$.

Next, we define for each index sequence π a term t_π in such a way that a principal in a protocol can easily verify whether or not π is a solution. Similarly as above, we set

$$t_\pi^0 = f(\bar{u}_\pi^0(c), c, \bar{v}_\pi^0(c))\ , \tag{17 a}$$

$$t_\pi^{j+1} = f(\bar{u}_\pi^{j+1}(c), t_\pi^j, \bar{v}_\pi^{j+1}(c))\ , \tag{17 b}$$

and $t_\pi = t_\pi^l$. The crucial property to observe is:
(ii) An index sequence π is a solution to the given instance of the PCP iff t_π matches $f(x, y, x)$.

Furthermore, there is a simple recursive criterion for checking whether a term t is identical to a term t_π for some π:
(iii) Let t be an arbitrary term. Then there exists an index sequence π such that $t = t_\pi$ iff one of (A) and (B) below is satisfied:

(A) There exists $i < m$ such that $t = f(\bar{u}_i(c), c, \bar{v}_i(c))$.

(B) There exists $i < m$ such that t matches $f(\bar{u}_i(x_1), f(x_1,x_2,x_3), \bar{v}_i(x_3))$, say with substitution σ, and $\sigma(f(x_1,x_2,x_3))$ is identical to $t_{\pi'}$ for some index sequence π'.

Reason. We first prove the direction from right to left. Clearly, if (A) is satisfied, then the index sequence i is a solution to the PCP instance. Now, assume (B) holds. Then we can take $\pi = \pi' i$. For the other direction, assume $t = t_\pi$ for some index sequence π. Then a simple induction on the length of π shows that (A) or (B) holds.

Whether (iii) holds can be "checked" by a selecting theory. The main idea is to traverse the tree top-down, checking in each step that some pair from the PCP instance was used to construct the current level of the tree.

The technical problem that arises is the restricted form of push clauses. They force us to make two passes over the tree: In the first pass, we only check the odd levels; in the second pass, we check the even levels.

The full selecting theory Φ we use is:

$$r_0(x) \to r(x) \ , \tag{18 a}$$
$$r_0(f(x_1, x_2, x_3)) \to r(x_2) \ , \tag{18 b}$$
$$r(f(\bar{u}_i(x_1), f(x_1, x_2, x_3), \bar{v}_i(x_3))) \to r(x_2) \ , \quad \text{for } i < m \ , \tag{18 c}$$
$$r(f(\bar{u}_i(c), c, \bar{v}_i(c))) \to \mathtt{I}(k) \ , \quad \text{for } i < m \ , \tag{18 d}$$
$$r(c) \to \mathtt{I}(\{\$\}_k) \ , \tag{18 e}$$

where r_0 and r are two push symbols. Clauses (18 a) and (18 b) start the two passes, and (18 d) and (18 e) check the success of the two passes, while (18 c) is used for the recursive descent in both passes.

A technically involved analysis of the above selecting theory shows:
(iv) For every ground term t, $r_0(t) \vdash_\Phi \mathtt{I}(\$)$ iff $t = t_\pi$ for some index sequence π.

Let $\mathcal{P} = (P, \Phi)$ be the protocol with exactly one principal $[\tau]$ where τ is given by

$$f(x_1, x_2, x_1) \to r_0(f(x_1, x_2, x_1)) \ . \tag{19}$$

Then we can state, using (ii) and (iv):
(v) $\mathcal{P}$ is insecure iff the PCP instance is solvable.

This describes the desired reduction. What remains is to show how we can get rid of the additional symbols in the signature.

Step 2. Reduction of the secrecy problem for protocols over the extended signature to the same problem over the standard signature. We use a simple encoding for terms over the extended signature, more precisely, we define the function $F\colon T(\Sigma', X) \to T(\Sigma, X)$ by:

$$F(c) = c \ , \tag{20 a}$$
$$F(x) = x \ , \tag{20 b}$$
$$F(a^1(t)) = \langle a, F(t)\rangle \ , \quad \text{for } a \in A \ , \tag{20 c}$$
$$F(f(t_0, t_1, t_2)) = \langle c_f, \langle t_0, \langle t_1, t_2\rangle\rangle\rangle \ , \tag{20 d}$$

where t, t_0, t_1, and t_2 are arbitrary terms from $T(\Sigma)$; the signature Σ contains no keys, but $a_0, \ldots, a_{n-1}$ as additional constants. In addition, we define Φ_F to consist of all rules of Φ, but where each argument t of a push symbol is replaced by $F(t)$. We then have:
(vi) For every $t \in T(\Sigma)$, $r(t) \vdash_{\Phi_F} \mathtt{I}(\$)$ iff there exists $t' \in T(\Sigma')$ such that $t' = F(t)$ and $t' \vdash_\Phi \mathtt{I}(\$)$.

Reason. The direction from right to left is trivial. For the other direction, one first proves that if $r(t) \vdash \mathtt{I}(\$)$, then $t = F(t')$ for some term t'. The rest is straightforward.

Finally, let $\mathcal{P}' = (P', \Phi_F)$ be the protocol with exactly one principal $[\tau]$ where τ is given by

$$F(f(x_1, x_2, x_1)) \to r(F(f(x_1, x_2, x_1))) \ . \tag{21}$$

Then:
(vii) $\mathcal{P}$ is insecure iff $\mathcal{P}'$ is insecure.

From (v) and (vii), we obtain the desired result.

Note that the above proof uses only one principal and that this principal consists of just one protocol step.

8. CONCLUSION

In this paper, we further investigated the decidability of the secrecy problem for recursive protocols. We extended Truderung's selecting theory model to include the ability to model the generation of new nonces and keys. We showed that in this model secrecy is decidable in double-exponential time and that secrecy in the original selecting theory model is in fact undecidable and only holds under the restrictions considered in the present paper. One open question is whether the developed decision procedure allows for practical implementations as it only exploits the fact that insecure recursive protocols have "small" attacks.

Acknowledgment.

We would like to thank Tomasz Truderung for helpful discussions on the selecting theory model and his proof of decidability. We also thank Christof Löding for helpful discussions on the proof of the undecidability result.

9. REFERENCES

[1] J. Bull and D. Otway. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, Malvern, UK, 1997.

[2] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997. release October, 1rst 2002.

[3] H. Comon-Lundh and V. Cortier. New Decidability Results for Fragments of First-order Logic and Application to Cryptographic Protocols. In *Proceedings of the 14th Internatioinal Conference on Rewriting Techniques and Applications (RTA 2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 148–164. Springer, 2003.

[4] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[5] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. Technical report, 2000. Available from http://www.counterpane.com/ipsec.pdf.

[6] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*, November 1998. RFC 2409.

[7] K. O. Kürtz. Selecting Theories and Nonce Generation for Recursive Protocols. Technical Report 0709, Institut für Informatik, CAU Kiel, Germany, 2007.

[8] R. Küsters. On the Decidability of Cryptographic Protocols with Open-ended Data Structures. In L. Brim, P. Jancar, M. Kretinsky, and A. Kucera, editors, *13th International Conference on Concurrency Theory (CONCUR 2002)*, volume 2421 of *Lecture Notes in Computer Science*, pages 515–530. Springer-Verlag, 2002.

[9] R. Küsters. On the Decidability of Cryptographic Protocols with Open-ended Data Structures. *International Journal of Information Security*, 4(1–2):49–70, 2005. Online publication appeared 2004. DOI: 10.1007/s10207-004-0050-z.

[10] R. Küsters and T. Wilke. Automata-based Analysis of Recursive Cryptographic Protocols. In V. Diekert and M. Habib, editors, *21st Symposium on Theoretical*

Aspects of Computer Science (STACS 2004), volume 2996 of *Lecture Notes in Computer Science*, pages 382–393. Springer-Verlag, 2004.

[11] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2000.

[12] J. Mongy. *Transformation de noyaux reconnaissables d'arbres. Forêts RATEG*. PhD thesis, LIF de Lille, Université des Sciences et Technologies de Lille, France, 1981.

[13] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, January 1987.

[14] R.Küsters and T. Truderung. On the Automatic Analysis of Recursive Security Protocols with XOR. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, volume 4393 of *Lecture Notes in Computer Science*. Springer, 2007.

[15] T. Truderung. Regular protocols and attacks with regular knowledge. In R. Nieuwenhuis, editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE 2005)*, volume 3328 of *Lecture Notes in Computer Science*, pages 377–391. Springer-Verlag, 2005.

[16] T. Truderung. Selecting theories and recursive protocols. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 217–232. Springer-Verlag, 2005.

[17] K. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *Proceedings of the 20th International Conference on Automated Deduction (CADE 2005)*, volume 3328 of *Lecture Notes in Computer Science*, pages 337–352. Springer-Verlag, 2005.

[18] J. Zhou. Fixing a security flaw in IKE protocols. *Electronic Letter*, 35(13):1072–1073, 1999.