Automata-based Analysis of Recursive Cryptographic Protocols*

Ralf Küsters[1] and Thomas Wilke[2]

[1] Stanford University
kuesters@theory.stanford.edu
[2] Christian-Albrechts-Universität zu Kiel
wilke@ti.informatik.uni-kiel.de

Abstract. Cryptographic protocols can be divided into (1) protocols where the protocol steps are simple from a computational point of view and can thus be modeled by simple means, for instance, single rewrite rules—we call these protocols *non-looping*—and (2) protocols, such as group protocols, where the protocol steps are complex and typically involve an iterative or recursive computation—we call them *recursive*. While much is known on the decidability of security for non-looping protocols, only little is known for recursive protocols. In this paper, we prove decidability of security (w.r.t. the standard Dolev-Yao intruder) for a core class of recursive protocols and undecidability for several extensions. The key ingredient of our protocol model are specifically designed tree transducers which work over infinite signatures and have the ability to generate new constants (which allow us to mimic key generation). The decidability result is based on an automata-theoretic construction which involves a new notion of regularity, designed to work well with the infinite signatures we use.

1 Introduction

In most cryptographic protocols, principals are described by a fixed sequence of what we call *receive-send actions*. When performing such an action, a principal receives a message from the environment and, after some internal computation, reacts by returning a message to the environment. Research on automatic protocol analysis [23, 3, 5, 19] has concentrated on protocols where a receive-send action can basically be described by a single rewrite rule of the form $t \to t'$: When receiving a message m, the message $\sigma(t')$ is returned as output provided that σ is the matcher for t and m, i.e., $\sigma(t) = m$. In other words, an input message is processed by applying the rewrite rule *once* on the top-level. We call receive-send actions of this kind and protocols based on such receive-send actions *non-looping*. It has been proved that for non-looping protocols when analyzed w.r.t. a finite number of receive-send actions and the standard Dolev-Yao intruder where the message size is not bounded, security (more precisely, secrecy) is decidable even when principals can perform equality tests on arbitrary

* This work was partially supported by PROCOPE. The first author was also supported by the Deutsche Forschungsgemeinschaft (DFG).

messages [23, 3, 5, 19], complex keys are allowed [23, 5, 19], and the free term algebra assumption is relaxed by algebraic properties of XOR and Diffie-Hellman Exponentiation [8, 12, 9].

The main question we are concerned with in this paper is in how far security is decidable for protocols where receive-send actions are complex and typically involve an iterative or recursive computation; we call such receive-send actions and protocols containing such actions *recursive*. The answer to this question is not at all obvious since protocol models for non-looping protocols do not capture recursive protocols and there are almost no decidability results for recursive protocols (see the related work).

To illustrate the kind of receive-send actions which are performed in recursive protocols, let us consider the key distribution server S of the Recursive Authentication (RA) Protocol [7] (see also Section 4). In this protocol, the server S needs to perform the following *recursive* receive-send action: The server S first receives an a priori unbounded sequence of requests of pairs of principals who want to share session keys. Then, S generates sessions keys, and finally sends a sequence of certificates (corresponding to the requests) containing the session keys. Receive-send actions of this kind are typical for group protocols, but also occur in protocols such as the Internet Key Exchange protocol (IKE)—see [18] for a description of some recursive protocols. As pointed out by Meadows [18] and illustrated in [25, 14], modeling recursion is security relevant.

A natural way to describe recursive receive-send actions is by tree transducers, which extend the class of transductions expressible by single rewrite rules (with linear left-hand side). More precisely, to study decidability, in Section 2 we introduce non-deterministic top-down tree transducers (TTACs) with look-ahead and epsilon transitions which work on a signature containing an *infinite* set of what we call *anonymous constants* (ACs), over which the TTACs has only very limited control. TTACs can generate new (anonymous) constants, a feature often needed to model recursive receive-send actions; in the RA protocol for instance, the key distribution server needs to generate (an a priori unbounded number of) session keys.

The main result of this paper is that i) security (for a finite number of receive-send actions, atomic keys, and the standard Dolev-Yao intruder where the message size is not bounded) is decidable if receive-send actions are modeled by TTACs (Section 5), and that ii) certain features of models for non-looping protocols cannot be added without losing decidability: As soon as TTACs are equipped with the ability to perform equality tests between arbitrary messages, complex keys are allowed, or the free term algebra assumption is relaxed by adding XOR or Diffie-Hellman Exponentiation security is undecidable (Section 6).

The undecidability results are obtained by reductions from Post's Correspondence Problem. The decidability result is obtained in two steps. First, we show that TTACs are powerful enough to simulate the intruder. This allows us to describe attacks as the composition of transducers. We can then reduce the security problem to the iterated pre-image word problem for a composition of TTACs, which we show to be decidable (Section 2.3): Given a term t, a "regular set" R of terms, and a sequence of TTACs, the *iterated pre-image word problem* asks whether on input t the composition of TTACs can produce an output in R. Here, "regular set" means the set of terms recognizable by a new kind of tree automata, *tree automata over signatures with anonymous constants* (TAACs), which can compare anonymous constants for equality.

See our technical report [16] for detailed proofs of all results presented here.

Related work. Recursive protocols have been analyzed manually [22] and semi-automatically using theorem provers or special purpose tools [21, 6, 17].

Decidability for recursive protocols has initially been investigated in a previous paper [15]. However, there are several significant differences to the present paper. Among others, in [15] *word* transducers, as opposed to *tree* transducers, were considered which do not allow generating new constants (e.g., session keys), a common cause for undecidability of security (see, e.g., [2] and references therein). Also, the proof techniques employed are completely different (see [16] for more details).

In various papers, automata-theoretic techniques have been applied to the analysis of cryptographic protocols (see, e.g., [2] and references therein). However, these works aim at non-looping protocols and do not seem to be applicable to recursive protocols in an obvious way. To the best of our knowledge, the work in [15] and the present work are the first to employ transducers (over infinite signatures) for protocol analysis. Automata and transducers over infinite signatures (although different from those considered here) have been studied in the context of type checking and type inference for XML queries with data values (see, e.g., [1, 20]).

Basic definitions and notation. If Σ is a signature, let Σ_n denote the set of symbols in Σ of arity n. The set of terms over Σ is denoted T_Σ. For a set C of constant symbols (symbols of arity 0) disjoint from Σ let $T_\Sigma(C) = T_{\Sigma\cup C}$. We fix an infinite supply X of variables among which we find x_0, x_1, x_2, ... For $n \geq 0$, we write T^n_Σ for the set of all terms in $T_\Sigma(\{x_0, \dots, x_{n-1}\})$. When $t \in T^n_\Sigma$ and $t_0, \dots, t_{n-1}$ are arbitrary terms, we write $t[t_0, \dots, t_{n-1}]$ for the term which is obtained from t by simultaneously substituting t_i for x_i, for every $i < n$. A term $t \in T^n_\Sigma$ is *linear* if every x_i with $i < n$ occurs exactly once in t. A substitution over Σ is a function $\sigma\colon T_\Sigma(X) \to T_\Sigma(X)$ such that for each term t, $\sigma(t)$ is obtained from t by simultaneously substituting $\sigma(x)$ for x for every $x \in X$. A subset τ of $T_\Sigma \times T_\Sigma$ is called a *transduction* over Σ. For $t \in T_\Sigma$, we define $\tau(t) = \{t' \mid (t, t') \in \tau\}$. If τ and τ' are transductions over Σ, then their *composition* $\tau \circ \tau'$ is defined as expected, where the composition is read from right to left. Given a transduction τ over Σ and a set $R \subseteq T_\Sigma$, the *pre-image* of R under τ is the set $\tau^{-1}(R) = \{t \mid \exists t'(t' \in R \wedge (t, t') \in \tau\}$.

2 Tree Automata and Transducers with Anonymous Constants

In this section we describe the models of tree automata and transducers that we use, completely independent of the application we have in mind, as they are of general interest. (See [11] for more information on tree automata and transducers.)

A pair (Σ, C) consisting of a finite signature Σ, the set of *regular symbols*, and an arbitrary *infinite* set C of constant symbols, the set of *anonymous constants*, disjoint from Σ is called a *signature with anonymous constants.* When we speak of a *term over* (Σ, C) we mean a term over $\Sigma \cup C$. In what follows, let $occ_C(t)$ ($occ_C(\mathcal{S})$) denote the set of elements from C that occur in the term t (the set of terms $\mathcal{S}$).

2.1 Tree Automata over Signatures with Anonymous Constants

Our tree automata are non-deterministic bottom-up tree automata that accept trees over signatures with anonymous constants; they have full control over the regular symbols but only very limited control over the anonymous constants. These automata are designed in such a way that they are powerful enough to recognize tree languages such as the one defined in Example 1 (which is needed for our cryptographic application) and such that they fit well with the tree transducers introduced in Section 2.2, in the sense that we can prove Theorem 1.

Formally, a *tree automaton (TAAC) over a signature* (Σ, C) *with anonymous constants* is a tuple $\mathbf{A} = (Q, q^d, q^s, \Delta, F)$ where Q is a non-empty finite *set of states,* $q^d \in Q$ is the *default state,* $q^s \in Q$ is the *selecting state,* Δ is a finite *set of transitions* as specified below, and $F \subseteq Q$ is a *set of final states.* The latter can be omitted; in this case, we speak of a *semi TAAC.*

There are two types of transitions: a *consuming transition* is of the form $f(q_0, \ldots, q_{n-1}) \to q$ where $f \in \Sigma_n$, $q, q_0, \ldots, q_{n-1} \in Q$; an *epsilon transition* is of the form $q' \to q$ where $q', q \in Q$.

Each TAAC over a signature with anonymous constants (Σ, C) defines a set of terms (also called trees) from $T_\Sigma(C)$. To describe this set, we view the set Q as a set of constants and introduce the notion of a permitted substitution. A *permitted substitution* σ is a function $\sigma\colon C \to \{q^d, q^s\}$ where at most one element of C, the *selected element,* gets assigned q^s and all the others get assigned q^d, the *default value.* For $t \in T_\Sigma(Q)$ (which does not contain anonymous constants), we define $[t]_\mathbf{A}$ inductively as follows: The set $[t]_\mathbf{A}$ is the smallest set such that if $t \in Q$, then $t \in [t]_\mathbf{A}$; if $t = f(t_0, \ldots, t_{n-1})$ and there exist $q_0, \ldots, q_{n-1}$ such that $f(q_0, \ldots, q_{n-1}) \to q \in \Delta$ and $q_i \in [t_i]_\mathbf{A}$ for every $i < n$, then $q \in [t]_\mathbf{A}$; if $q \in [t]_\mathbf{A}$ and $q \to q' \in \Delta$, then $q' \in [t]_\mathbf{A}$. For every term $t \in T_\Sigma(C \cup Q)$ (which may contain anonymous constants), the set $[t]_\mathbf{A}$ of states which the automaton reaches after having read the term t is defined to be the union of all sets $[\sigma(t)]_\mathbf{A}$ where the union is taken over all permitted substitutions σ. Now, the *tree language recognized* by $\mathbf{A}$ is the language $T(\mathbf{A}) = \{t \in T_\Sigma(C) \mid F \cap [t]_\mathbf{A} \neq \emptyset\}$. We say a tree language over (Σ, C) is *TAAC recognizable* over (Σ, C) if it is recognized by some TAAC over (Σ, C).

Example 1. Assume $\Sigma_2 = \{f\}$ and $\Sigma_i = \emptyset$ for every $i \neq 2$. Let $T_= = \{f(c, c) \mid c \in C\}$. This language is recognized by a TAAC with only three states, say q_0, q_1, and q_2. We choose $q^d = q_0$, $q^s = q_1$, $F = \{q_2\}$, and we have only one transition, namely $f(q_1, q_1) \to q_2$.

We will also use what we call a *weak TAAC (WTAA)* which does not have a selecting state, i.e., the default state is assigned to all anonymous constants. WTAAs are really weaker because it is easy to see that, for instance, $T_=$ is not WTAAC recognizable over (Σ, C). We can prove the following basic properties of TAACs and WTAACs.

Lemma 1. *Let (Σ, C) be a signature with an infinite set C of anonymous constants. The set of TAAC recognizable tree languages over (Σ, C) is closed under union. It is closed under intersection/complementation over (Σ, C) iff $\Sigma = \Sigma_0 \cup \Sigma_1$. The word problem and the emptiness problem for TAACs are decidable. The set of WTAAC recognizable tree languages over (Σ, C) is closed under union, intersection, and complementation.*

2.2 Tree Transducers over Signatures with Anonymous Constants

As mentioned in the introduction, we use non-deterministic top-down tree transducers with epsilon transitions which have the following specific features: a WTAAC look-ahead; generation of new (!) anonymous constants; a register for one anonymous constant.

To define our transducers we need some notation. We fix a signature (Σ, C) with anonymous constants and a finite set S of states, whose elements we view as *binary* symbols. We assume that we are given a set $V = \{v_R, v_N\}$ of two variables for anonymous constants: v_R represents the aforementioned register, v_N refers to a newly generated anonymous constant. A *state term* is of the form $s(z, t)$ for $s \in S$, $z \in V \cup C \cup \{*\}$, and $t \in T_\Sigma(C \cup X)$. The term t is then called the *core term* of this term. If z belongs to some set $D \subseteq V \cup C \cup \{*\}$, then we say $s(z, t)$ is a *D-state term*. Intuitively, a state term of the form $s(*, t)$ or $s(c, t)$ with $c \in C$ is part of a configuration of a transducer and means that the transducer is about to read t starting in state s where the register does not store a value or stores the anonymous constant c, respectively. To describe transitions we use state terms of the form $s(v_R, t)$, $s(v_N, t)$, and again $s(*, t)$, but not $s(c, t)$ (see below).

Formally, a *tree transducer (TTAC) over a signature (Σ, C) with anonymous constants* is a tuple $\mathbf{T} = (S, I, \mathbf{A}, \Gamma)$ where S is a finite set of *states*, $I \subseteq S$ is a set of *inital states*, $\mathbf{A}$ is a semi WTAAC over (Σ, C), and Γ is a finite set of transitions. A *transition* is of the form

$$s(z, t) \to^q t'[v_R, v_N, t'_0, \ldots, t'_{r-1}] \tag{1}$$

where $q \in Q$ is the *look-ahead*, $s(z, t)$ is a $\{v_R, *\}$-state term (recall that this means that $z = v_R$ or $z = *$) with $t \in T^n_\Sigma$ and t linear, $t' \in T^{r+2}_\Sigma$ (not necessarily linear), and each t'_i is either a variable x_j with $j < n$ or a $\{z, v_N, *\}$-state term with the core term being a subterm of t. If $z = *$, we require that v_R does not occur in $t'[v_R, v_N, t'_0, \ldots, t'_{r-1}]$. If v_N occurs in this term, the transition is called *generative* and *non-generative* otherwise.

The computation the TTAC carries out is described by a sequence of rewrite steps. The corresponding rewrite relation $\vdash_U$ is defined w.r.t. a subset $U \subseteq C$ of anonymous constants to ensure that newly generated constants do not belong to U. Later, U will be the set of anonymous constants in the input term, which then ensures that the anonymous constants generated by the TTAC are different from those occurring in the input. To define $\vdash_U$, assume we are given a term $u_0 = u_1[s(c, u_2)]$ where $c \in C$, a term $u_2 = t[t_0, \ldots, t_{n-1}] \in T_\Sigma(C)$, a transition T as in (1) with $z = v_R$, and assume u_1 is linear. Let σ be the substitution defined by $\sigma(x_i) = t_i$. Then, for every $c' \in C \setminus (occ_C(u_0) \cup U)$, if $q \in [u_2]_{\mathbf{A}}$ we define $u_0 \vdash_U u_1[t'[c, c', \sigma(t'_0), \ldots, \sigma(t'_{r-1})]]$. (Observe that if T is non-generative, v_N and U are irrelevant.) Note that the newly generated anonymous constant c' does not occur in U and the output term computed so far. The rewrite step in case $u_0 = u_1[s(*, u_2)]$ is defined analogously where transitions as in (1) can only be applied if $z = *$.

Let $\vdash_U^*$ denote the reflexive transitive closure of $\vdash_U$. The transduction over (Σ, C) defined by the TTAC is

$$\tau_{\mathbf{T}} = \{(t, t') \in T_\Sigma(C) \times T_\Sigma(C) \mid \exists s (s \in I \wedge s(*, t) \vdash^*_{occ_C(t)} t')\} \ .$$

A transduction τ on (Σ, C) is called *TTAC realizable* if there exists a TTAC $\mathbf{T}$ such that $\tau_{\mathbf{T}} = \tau$. Just as for classical non-deterministic top-down tree transducers [11], it is easy to see that the set of TTAC realizable transductions is not closed under composition.

2.3 The Iterated Pre-image Word Problem

The iterated pre-image word problem is defined as follows:

IteratedPreImage. Given a term t over (Σ, C), a TAAC $\mathbf{B}$ over (Σ, C), and a sequence of TTACs $\mathbf{T_0}, \ldots, \mathbf{T_{l-1}}$ over (Σ, C) with $\tau = \tau_{T_0} \circ \cdots \circ \tau_{T_{l-1}}$, decide whether $t \in \tau^{-1}(T(\mathbf{B}))$.

The key for proving decidability of this problem is:

Theorem 1. *The pre-image of a TAAC recognizable tree language under a TTAC realizable transduction is a TAAC recognizable tree language. Moreover, an appropriate TAAC can be constructed effectively.*

To prove this theorem, given a TAAC $\mathbf{B}$ and a TTAC $\mathbf{T}$, we provide an exponential time construction of a TAAC recognizing $\tau_{\mathbf{T}}^{-1}(T(\mathbf{B}))$. Using Theorem 1 and Lemma 1 (decidability of the word problem), we obtain:

Corollary 1. IteratedPreImage *is decidable.*

3 The Tree Transducer-based Protocol Model

We now define our tree transducer-based protocol model by specifying messages, the intruder, protocols, and attacks. As mentioned in the introduction, the main difference between the model presented here and decidable models for non-looping protocols is the way receive-send actions are described—instead of single rewrite rules, we use TTACs. These transducers have two important features necessary to model recursive receive-send actions, but missing in decidable models for non-looping protocols: First, they allow to apply a set of rewrite rules recursively to a term. Second, they allow to generate new constants—a feature not necessary for non-looping protocols when analyzed w.r.t. a finite number of receive-send actions.

Due to the space limit, the exposition below is slightly simplified. For details, the reader is referred to [16].

3.1 Messages

The definition of messages we use here is rather standard, except that we allow an infinite number of (anonymous) constants. As mentioned, we assume keys to be atomic.

More precisely, messages are defined as terms over the signature $(\Sigma_\mathcal{A}, \mathcal{C})$ with anonymous constants. The set $\mathcal{C}$ is some *infinite* set of anonymous constants, which in this paper will be used to model session keys (Section 4). The finite signature $\Sigma_\mathcal{A}$ is defined relatively to a *finite* set $\mathcal{A}$ of constants, the set of *atomic messages*, which may for instance contain principal names and (long-term) keys. It also contains a subset $\mathcal{K} \subseteq \mathcal{A}$ of public and private keys which is equipped with a bijective mapping $\cdot^{-1}$ assigning to a public (private) key $k \in \mathcal{K}$ its corresponding private (public) key $k^{-1} \in \mathcal{K}$. Now, $\Sigma_\mathcal{A}$ denotes the (finite) signature consisting of the constants $\mathcal{A}$, the unary symbols $\mathtt{hash}_a$ (*keyed hash*) and $\mathtt{enc}_a^s$ (*symmetric encryption*) for every $a \in \mathcal{A}$, $\mathtt{enc}_k^a$ (*asymmetric encryption*) for every $k \in \mathcal{K}$, and the binary symbol $\langle\rangle$ (*pairing*). Instead of $\langle\rangle(t, t')$ we write $\langle t, t'\rangle$. We point out that $\mathtt{hash}_a(m)$ shall represent the keyed hash of m under the key a plus m itself. Note that anonymous constants are not allowed as keys (see also Section 3.4 and 7). The set of *messages* over $(\Sigma_\mathcal{A}, \mathcal{C})$ is denoted $\mathcal{M} = T_{\Sigma_\mathcal{A}}(\mathcal{C})$.

3.2 Receive-send Actions, Principals, and Protocols

A *receive-send action* is a TTAC over $(\Sigma_\mathcal{A}, \mathcal{C})$. Roughly speaking, a *principal* is defined to be a finite sequence of receive-send actions, where the last action may or may not be marked to be what we call a *challenge output action* (see [16] for a precise definition of principals). The purpose of challenge output actions is explained below. A protocol is a tuple consisting of a finite family of principals and a finite set $\mathcal{S} \subseteq \mathcal{M}$, the *initial intruder knowledge*.

3.3 The Intruder

As in the case of models for non-looping protocols, our intruder model is based on the well-known and widely used Dolev-Yao intruder [13]. That is, an intruder has complete control over the network and can derive new messages from his current knowledge by composing, decomposing, encrypting, decrypting, and hashing messages. We do not impose any restrictions on the size of messages.

The (possibly infinite) set of messages $\mathsf{d}(\mathcal{S})$ the intruder can derive from some set $\mathcal{S} \subseteq \mathcal{M}$ is the smallest set satisfying the following conditions: $\mathcal{S} \subseteq \mathsf{d}(\mathcal{S})$; if $\langle m, m' \rangle \in \mathsf{d}(\mathcal{S})$, then $m, m' \in \mathsf{d}(\mathcal{S})$ (decomposition); if $\mathtt{enc}^s_a(m) \in \mathsf{d}(\mathcal{S})$ and $a \in \mathsf{d}(\mathcal{S})$, then $m \in \mathsf{d}(\mathcal{S})$ (symmetric decryption); if $\mathtt{enc}^a_k(m) \in \mathsf{d}(\mathcal{S})$ and $k^{-1} \in \mathsf{d}(\mathcal{S})$, then $m \in \mathsf{d}(\mathcal{S})$ (asymmetric decryption); if $\mathtt{hash}_a(m) \in \mathsf{d}(\mathcal{S})$, then $m \in \mathsf{d}(\mathcal{S})$ (obtaining hashed messages); if $m, m' \in \mathsf{d}(\mathcal{S})$, then $\langle m, m' \rangle \in \mathsf{d}(\mathcal{S})$ (composition); if $m \in \mathsf{d}(\mathcal{S})$ and $a \in \mathcal{A} \cap \mathsf{d}(\mathcal{S})$, then $\mathtt{enc}^s_a(m), \mathtt{hash}_a(m) \in \mathsf{d}(\mathcal{S})$ (symmetric encryption and keyed hash); and if $m \in \mathsf{d}(\mathcal{S})$ and $k \in \mathcal{K} \cap \mathsf{d}(\mathcal{S})$, then $\mathtt{enc}^a_k(m) \in \mathsf{d}(\mathcal{S})$ (asymmetric encryption).

We note that although principals have the ability to generate new (anonymous) constants, as they are defined in terms of TTACs, for the intruder adding this ability is not necessary since it would *not* increase his power to attack protocols (see [16] for more details).

3.4 Attacks on Protocols

In a (successful) attack on a protocol P, the intruder chooses an interleaving $\mathbf{T}_0, \ldots, \mathbf{T}_{l-1}$ of the receive-send actions of the principals in P (i.e., a total ordering of these receive-send actions) in such a way that i) the last receive-send action $\mathbf{T}_{l-1}$ in this interleaving is a challenge output action, ii) the intruder can produce the input m_i for the receive-send actions, and iii) from the messages m'_i returned by the receive-send actions and his initial knowledge $\mathcal{S}$ he can derive a secret message. The secret message m'_{l-1} the intruder tries to derive is a regular or anonymous constant determined by the challenge output action $\mathbf{T}_{l-1}$ and it is presented to the intruder as a challenge but *not* added to his knowledge. In the following definition of attack, due to space limitations and for simplicity of notation, in this extended abstract we assume that the interleaving of receive-send actions is given beforehand. Since the number of possible interleavings is finite, from a decidability point of view this is not a restriction (see [16] for a full definition of attack). The second condition in the following definition ensures that new anonymous constants generated in the ith receive-send action are also new w.r.t. the knowledge of the intruder before the ith action is performed.

ATTACK. *Given a finite set $\mathcal{S} \subseteq \mathcal{M}$ (the initial intruder knowledge), TTACs $\mathbf{T}_0, \ldots, \mathbf{T}_{l-1}$ (the interleaving of receive-send actions) with $\mathbf{T}_i = (S_i, I_i, \mathbf{A}_i, \Gamma_i)$ for $i < l$, decide whether there exist messages $m_i, m'_i \in \mathcal{M}$, $i < l$, such that*

1. $(m_i, m'_i) \in \tau_{\mathbf{T}_i}$ *for every* $i < l$,
2. $(occ_{\mathcal{C}}(m'_i) \setminus occ_{\mathcal{C}}(m_i)) \cap occ_{\mathcal{C}}(\mathcal{S}_i) = \emptyset$ *for every* $i < l$,
3. $m_i \in \mathsf{d}(\mathcal{S}_i)$ *for every* $i < l$, *and*
4. $m'_{l-1} \in \mathsf{d}(\mathcal{S}_{l-1}) \cap (\mathcal{A} \cup \mathcal{C})$. *(Can the intruder derive the challenge?)*

where $\mathcal{S}_i = \mathcal{S} \cup \{m'_0, \ldots, m'_{i-1}\}$ is the intruder's knowledge before the ith receive-send action is performed.

We write $(\mathcal{S}, \mathbf{T}_0, \ldots, \mathbf{T}_{l-1}) \in$ ATTACK if all the above conditions are satisfied.

The use of challenge output actions, as presented above, allows to determine secrets dynamically, depending on the protocol run. This is for example needed when asking whether the intruder is able to derive a session key (an anonymous constant, which may change from one protocol run to another) generated by a key distribution server. Challenge output actions are somewhat related to the way security is defined in computational models for key distribution protocols where at the end of an attack, the intruder is presented a string for which he needs to decide whether it is an actual session key or just some random string [4]. In [16], we discuss alternatives to challenge output actions.

4 Modeling Recursive Cryptographic Protocols

To illustrate the TTAC-based protocol model, we now present a formal description of the key distribution server, called S in the following, of the Recursive Authentication (RA) Protocol [7] (see [16] for a complete description and a more detailed account of the RA protocol). In what follows, we abbreviate messages of the form $\langle m_0, \ldots, \langle m_{n-1}, m_n \rangle \cdots \rangle$ by $m_0, \ldots, m_n$.

The server S shares a long-term (symmetric) key with every principal and performs only one (recursive) receive-send action in a protocol run. In this receive-send action, S receives an a priori unbounded sequence of requests of pairs of principals who want to obtain session keys for secure communication and has to generate certificates for the principals containing the session keys. An example of the kind of message S receives is

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

where N_c, N_b, and N_a are nonces generated by C, B, and A respectively, and K_c, K_b, and K_a are the long-term keys shared between the server S and the principals C, B, and A, respectively. The above message consists of three requests and indicates that C wants to share a session key with S, B with C, and A with B. The symbol "$-$" marks the end of the sequence of requests. It is important to note that messages sent to S may contain an arbitrary number of requests—which must be processed by S recursively. Now, given m, S processes the requests starting from the outermost. First, S generates two certificates for C, namely, $\mathtt{enc}^s_{K_c}(K_{cs}, C, N_c)$ and $\mathtt{enc}^s_{K_c}(K_{cb}, B, N_c)$, where K_{cs} and K_{cb} are session keys generated by S and intended to be used by C for communication with S and B, respectively. In the same way, certificates for B and A are generated, where A only obtains one certificate (containing the session key for communication with B).

We now describe S by the TTAC $\mathbf{T}_S$. Let $P_0, \ldots, P_n$ be the principals participating in the RA protocol. We assume that $P_n = S$ is the server. Every P_i, $i < n$, shares a long-term key K_i with S. The transducer $\mathbf{T}_S$ has two states, start and read, and does not need a look-ahead—we will need a look-ahead to model the intruder in terms of a TTAC (Section 5). In state start, the initial state, $\mathbf{T_S}$ checks whether the first request is addressed to S and initializes the process of reading the requests by generating one session key which is stored in the register. In state read, the requests are processed. In this phase, the register is used to store a session key while moving from one request to the next.

The transitions of $\mathbf{T}_S$ are specified as follows:

$$\begin{aligned}
\mathsf{start}(*, \mathtt{hash}_{K_i}(P_i, P_n, x_0, x_1)) &\rightarrow \mathsf{read}(v_N, \mathtt{hash}_{K_i}(P_i, P_n, x_0, x_1))\\
\mathsf{read}(v_R, \mathtt{hash}_{K_i}(P_i, P_j, x_0, -)) &\rightarrow \mathtt{enc}^s_{K_i}(v_R, P_j, x_0)\\
\mathsf{read}(v_R, \mathtt{hash}_{K_i}(P_i, P_j, x_0, \mathtt{hash}_{K_{i'}}(P_{i'}, P_i, x_1, x_2))) &\rightarrow \mathtt{enc}^s_{K_i}(v_R, P_j, x_0),\\
&\ \mathtt{enc}^s_{K_i}(v_N, P_{i'}, x_0),\\
&\ \mathsf{read}(v_N, \mathtt{hash}_{K_{i'}}(P_{i'}, P_i, x_1, x_2))
\end{aligned}$$

where $i, i', j \leq n$ and x_0, x_1, x_2 are variables which take arbitrary messages, and v_R and v_N are the variables for the register and the new anonymous constants, respectively.

5 The Decidability Result

The main result of this section is the following:

Theorem 2. ATTACK *is decidable.*

The proof of this theorem proceeds in two steps. In the first step, we show that the intruder can be simulated by a TTAC, which we call $\mathbf{T}_{der}$. We point out that for the construction of $\mathbf{T}_{der}$ the use of a look-ahead is necessary to gather all information about which keys can be accessed in a given message. In the second step of the proof of Theorem 2, we describe attacks as composition of transducers $\mathbf{T}_{der}, \mathbf{T}_{l-1}, \mathbf{T}_{der}, \mathbf{T}_{l-2}, \mathbf{T}_{der}, \cdots \mathbf{T}_{der}, \mathbf{T}_0, \mathbf{T}_{der}$ (applied from right to left). More accurately, these transducers need to be slightly modified to pass on the intruder's knowledge from one transducer to the next. The (slightly modified) transducer $\mathbf{T}_{l-1}$ will produce a pair $\langle m, m'_{l-1}\rangle$ where m represents the intruder's current knowledge $\mathcal{S}_{l-1}$ and m'_{l-1} is the challenge. Given $\langle m, m'_{l-1}\rangle$, the transducer $\mathbf{T}_{der}$ (again slightly modified in a similar fashion as above) will now try to transform m into m'_{l-1}, i.e., try to derive the challenge from m, without using m'_{l-1}. In other words, in the last step $\mathbf{T}_{der}$ tries to produce a pair in the set $R = \{\langle a, a\rangle \mid a \in \mathcal{A}\} \cup \{\langle c, c\rangle \mid c \in \mathcal{C}\}$. Using Example 1, it is easy to see that R is TAAC recognizable over $(\Sigma_{\mathcal{A}}, \mathcal{C})$. The following lemma formalizes the reduction from ATTACK to ITERATEDPREIMAGE where τ is the transduction obtained from the composition of transducers just described and $m_{\mathcal{S}} = \langle u_0, \langle \cdots \langle u_{n-2}, u_{n-1}\rangle \cdots\rangle$ represents the intruder's initial knowledge $\mathcal{S} = \{u_0, \ldots, u_{n-1}\}$. This lemma, together with Corollary 1, immediately implies Theorem 2.

Lemma 2. *We have* $(\mathcal{S}, \mathbf{T_0}, \ldots, \mathbf{T_{l-1}}) \in$ ATTACK *if and only if* $m_{\mathcal{S}} \in \tau^{-1}(R)$.

6 Adding Features of Models for Non-looping Protocols and Undecidability Results

In the TTAC-based protocol model as introduced in Section 3, many non-looping protocols can be analyzed with the same precision as in decidable models for non-looping protocols with atomic keys (see, e.g., [3]). More precisely, this is the case for protocols where a) the receive-send actions can be described by rewrite rules with linear left-hand side, since TTACs can simulate *all* such rewrite rules, and b) only a finite amount of information needs to be conveyed from one receive-send action to the next. This includes for instance many of the protocols in the Clark-Jacobs library [10] (see [16] for a formal TTAC-based model of the Needham-Schroeder Public Key Protocol).

However, some features present in decidable models for non-looping protocols are missing in the TTAC-based protocol model: i) Equality tests for messages of arbitrary size, which are possible when left-hand sides of rewrite rules may be non-linear (this corresponds to allowing non-linear left-hand sides in transitions of TTACs) or arbitrary messages can be conveyed from one receive-send action to another and can then be compared with other messages [3, 23, 19, 5]; ii) complex keys, i.e., keys that may be arbitrary messages [23, 19, 5]; and iii) relaxing the free term algebra assumption by adding the XOR operator [8, 12] or Diffie-Hellman Exponentiation [9]. The main result of this section is that these features cannot be added without losing decidability (see [16] for a formal statement and the proof, in which we present reductions from Post's Correspondence Problem):

Theorem 3. ATTACK *is undecidable when one (or more) of the above features is added to the TTAC-based protocol model.*

7 Conclusion

The main goal of this paper was to shed light on the feasibility of automatic analysis of recursive cryptographic protocols. The results obtained here trace a fairly tight boundary of the decidability of security for such protocols. To obtain our results we introduced tree automata (TAACs) and transducers (TTACs) over signatures with an infinite set of (anonymous) constants and proved that for TTACs the iterated pre-image word problem is decidable. We believe that the study of TAACs and TTACs started here is of independent interest.

Our decision procedure for finding attacks on protocols is non-elementary and the problem can easily be seen to be EXPTIME-hard. Thus, one open problem is to establish tight complexity bounds. While so far we do not allow anonymous constants as keys, this would be an interesting extension of our model. In this paper, we have identified the computation of pre-images as a means to analyze protocols. It is worthwhile to investigate to what extent this method is practical and whether it could be an altnerative to constraint solving approaches usually employed for the analysis of (non-looping) protocols.

References

1. N. Alon, T. Milo, F. Neven, D. Suciu, and V. Vianu. XML with Data Values: Typechecking Revisited. In *JCSS 66(4): 688-727 (2003).*
2. R.M. Amadio and W. Charatonik. On Name Generation and Set-Based Analysis in the Dolev-Yao Model. In *CONCUR 2002.*

3. R.M. Amadio, D. Lugiez, and V. Vanackere. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.
4. M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *STOC'95*, pages 57–66. ACM, 1995.
5. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *ICALP 2001*, pages 667–681. Springer, 2001.
6. J. Bryans and S.A. Schneider. CSP, PVS, and a Recursive Authentication Protocol. In *DIMACS Workshop on Formal Verification of Security Protocols*, 1997.
7. J.A. Bull and D.J. Otway. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, Malvern, UK, 1997.
8. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *LICS 2003*.
9. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FSTTCS 2003*.
10. J. Clark and J. Jacob. *A Survey of Authentication Protocol Literature*, 1997. Web Draft Version 1.0 available from http://citeseer.nj.nec.com/.
11. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications, 1997. Available from http://www.grappa.univ-lille3.fr/tata.
12. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS 2003*.
13. D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
14. N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. Technical report, 2000. Available from http://www.counterpane.com/ipsec.pdf.
15. R. Küsters. On the decidability of cryptographic protocols with open-ended data structures. In *CONCUR 2002*, LNCS 2421, pages 515–530. Springer, 2002.
16. R. Küsters and Th. Wilke. Automata-based Analysis of Recursive Cryptographic Protocols. Technical Report IFI 0311, CAU Kiel, 2003. Available from http://www.informatik.uni-kiel.de/reports/2003/0311.html
17. C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In *WITS 2000*.
18. C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2000.
19. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS 2001*, pages 166–175. ACM Press, 2001.
20. F. Neven, Th. Schwentick, and V. Vianu. Towards regular languages over infinite alphabets. In *MFCS 2001*, LNCS 2136, pages 560–572. Springer, 2001.
21. L.C. Paulson. Mechanized Proofs for a Recursive Authentication Protocol. In *CSFW-10*, pages 84–95. IEEE Computer Society Press, 1997.
22. O. Pereira and J.-J. Quisquater. A Security Analysis of the Cliques Protocols Suites. In *CSFW-14*, pages 73–81. IEEE Computer Society Press, 2001.
23. M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *CSFW-14*, pages 174–190. 2001.
24. H. Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52(2), 1994.
25. J. Zhou. Fixing a security flaw in IKE protocols. *Electronic Letter*, 35(13):1072–1073, 1999.