INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Tree Transducer-based Analysis of Cryptographic Protocols

Ralf Küsters

Bericht Nr. 0301
January 2003

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Tree Transducer-based Analysis of Cryptographic Protocols

Ralf Küsters

Bericht Nr. 0301
January 2003

e-mail: kuesters@ti.informatik.uni-kiel.de

Tree Transducer-based Analysis of Cryptographic Protocols

Ralf Küsters
Institut für Informatik und Praktische Mathematik
CAU Kiel, Germany
kuesters@ti.informatik.uni-kiel.de

Abstract

In many cryptographic protocols, the actions performed by principals are iterative processes. However, in contrast to other classes of protocols, only very little is known about deciding the security of such protocols. To analyze decidability for these protocols, we propose a protocol model in which the actions of principals are described by tree transducers with regular look-ahead. The main result is that security in this model is decidable and EXPTIME-hard in presence of the standard Dolev-Yao intruder.

1 Introduction

Formal analysis has proved very successful in finding flaws in many published cryptographic protocols [6]. Even fully automatic analysis of such protocols is possible, based on models for which security is decidable (see [18, 8] for an overview of the different approaches, decidability, and complexity theoretic results).

In most cryptographic protocol, principals are described by a fixed sequence of what we will call *receive-send actions*. When performing a receive-send 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 has concentrated on protocols where receive-send actions can basically be described by single rewrite rules of the form $t \to t'$: On receiving a message, i.e., a ground term m, t is matched against m yielding a substitution σ (if any), and $\sigma(t')$ is returned as output. That is, an input message is processed by applying the rewrite rule *once* (on top-level). In the literature on cryptographic protocol analysis such transformations are described in different ways [23, 1, 3, 19, 7]. We will call receive-send actions of this kind and protocols based on such receive-send actions *non-looping*.

In this paper, a protocol model is proposed in which receive-send actions are described by (non-deterministic) tree transducers with regular look-ahead (TTLAs). Roughly speaking, such automata, which were first introduced by Engelfriet [10], are given by a set of rewrite rules of a certain kind, and they transform input messages to output messages by *iteratively* applying the rewrite rules to the input message. Thus, in contrast to models for non-looping protocols, in our model we can describe what we will call *iterative protocols*, i.e., protocols with receive-send actions that perform certain iterative processes on input messages (see Section 8 for a brief comparison between the different models).

The main result of this paper is that in the tree transducer-based protocol model, security — more precisely, secrecy — is decidable and EXPTIME-hard. The proof technique devised is very different from the one for non-looping protocols, because, besides the intruder, iteration is an additional source of the infiniteness of the protocol model. To demonstrate the limits of automatic analysis of iterative protocols, we show that certain extensions of TTLAs lead to undecidability of security.

Many iterative protocols have been published — see [18] for a description of some of them, including the Internet Key Exchange protocol (IKE) and different group protocols —, and as pointed out by Meadows [18] and illustrated in [25, 11], modeling iteration is security relevant. One example of an iterative protocol is the recursive authentication protocol (RA protocol) [5], a group protocol. In the RA protocol a key distribution server receives, in one receive-send action, an a priori unbounded sequence of request messages, where each request message contains a pair of principals who want to share session keys. From this sequence, the server needs to generate a corresponding sequence of certificates containing the session keys. This receive-send action is an iterative process, which can be described by a TTLA but is out of the scope of models for non-looping protocols. Appendix D.1 provides a detailed description of the RA protocol and a formalization of this protocol in our tree transducer-based protocol model.

Related work. So far, only a few contributions on the analysis of iterative protocols exist, mostly semi-automatic (using theorem provers or special purpose tools) and manual analysis has been carried out [21, 4, 17, 22].

Decidability issues for iterative protocols have initially been investigated in a previous paper [15], in which *word* transducers are used as a first simple device to describe iterative processes. In this setting, messages are interpreted as words, and thus, only messages of a certain simple structure can be parsed. By using TTLAs, one obtains a more natural and a more powerful formalism, which is applicable to a wider class of protocols (see Section 8) and which provides a clearer pictures of the differences between iterative and non-looping protocols. Similar to the present work, in [15] a pumping argument based on different quasi-orderings was applied to show decidability of security.

Structure of the paper. In the following section, some basic notions are recalled. In Section 3, tree transducers are introduced and Section 4 presents the tree transducer-based protocol model. The undecidability results are proved in Section 5, the decidability result in Section 6, and Section 7 contains the proof of the complexity lower bound. A brief comparison between the tree transducer-based protocol model and models for non-looping protocols is provided in Section 8. We conclude in Section 9. The appendix contains a description of the recursive authentication protocol and its formulation in terms of the tree transducer-based model. Also, some of the proofs are moved to the appendix.

2 Basic Notions

A *ranked alphabet* is a tuple (Σ, arity) where Σ is a finite set of (function) symbols and arity assigns to every symbol in Σ a non-negative integer, its *arity*. Often we simply write Σ if the arity of the symbols in Σ is clear from the context. Symbols of arity n are called n-ary symbols. As usual, symbols of arity 0, 1, and 2, are refered to as *constants*, *unary*, and *binary* symbols, respectively. By Σ_n we denote the set of all n-ary symbols in Σ.

Let $\mathcal{V}$ be a set of symbols disjoint from Σ, the set of *variables*. The set $\mathcal{T}(\Sigma, \mathcal{V})$ of *terms* over the ranked alphabet Σ and the set of variables $\mathcal{V}$ is the smallest set satisfying:

- $\mathcal{V} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$,
- If $n \geq 0$, $f \in \Sigma_n$, and $t_0, \ldots, t_{n-1} \in \mathcal{T}(\Sigma, \mathcal{V})$, then $f(t_0, \ldots, t_{n-1}) \in \mathcal{T}(\Sigma, \mathcal{V})$.

We define $Var(t)$ to be the set of variables occurring in t. If $Var(t) = \emptyset$, t is called *ground term*. We define $\mathcal{T}(\Sigma)$ to be the set of ground terms over Σ. A term t is called *linear* if every variable in t occurs at most once in t. A linear term C containing n distinct variables is called an *(n-ary) context*. The set of n-ary contexts over the ranked alphabet Σ is denoted by $\mathcal{C}^n(\Sigma)$.

We write $t(v_0, \ldots, v_{n-1})$ to say that t is a term over the variables $v_0, \ldots, v_{n-1}$, i.e., $t \in \mathcal{T}(\Sigma, \{v_0, \ldots, v_{n-1}\})$. For terms $t_0, \ldots, t_{n-1}$, the expression $t[t_0, \ldots, t_{n-1}]$ denotes the term obtained by simultaneously replacing every variable v_i in t by t_i.

With ω we denote the set of non-negative integers and ω^* is the set of finite strings over ω. The empty string is ε. With $\leq$ we denote the prefix ordering on ω^*, i.e., for $v, w \in \omega^*$, $v \leq w$ iff there exists $v' \in \omega^*$ such that $vv' = w$, where vv' denotes the concatenation of v and v'. In this case, v is called a *prefix* of w. A set $S \subseteq \omega^*$ is called *prefix closed* if $v \in S$ implies that every prefix of v belongs to S.

A term t can be seen as a mapping from a non-empty, finite, and prefix closed set $\mathcal{S} \subseteq \omega^*$ into Σ such that if $t(\pi) \in \Sigma_n$ for some $n \geq 0$, then $\{i \mid \pi i \in \mathcal{S}$ and $i \geq 0\} = \{0 \ldots, n-1\}$, and if $t(\pi) \in \mathcal{V}$, then $\{i \mid \pi i \in \mathcal{S}$ and $i \geq 0\} = \emptyset$. We call $\mathcal{S}$ the *set of positions* of t and denote this set by $\mathcal{POS}(t)$.

For a term t and $\pi \in \mathcal{POS}(t)$, $t|_\pi$ shall denote the *subterm* of t at position π, i.e., $\mathcal{POS}(t|_\pi) := \{\pi' \mid \pi\pi' \in \mathcal{POS}(t)\}$ and for every $\pi' \in \mathcal{POS}(t|_\pi)$, $t|_\pi(\pi') := t(\pi\pi')$.

If $t(v_0, \ldots, v_{n-1})$ is a linear term, then $\mathsf{pos}_t(v_i)$ shall denote the (unique) position of v_i in t.

The *depth* $\mathsf{depth}(t)$ *of a term* t is defined as follows:

- $\mathsf{depth}(a) := 0$ for some constant or a variable a,
- $\mathsf{depth}(f(t_0, \ldots, t_{n-1})) := 1 + \mathsf{max}\{\mathsf{depth}(t_0), \ldots, \mathsf{depth}(t_{n-1})\}$.

This definition extends to finite subset of terms $\mathcal{S} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ as follows:

$$\mathsf{depth}(\mathcal{S}) := \mathsf{max}\{\mathsf{depth}(t) \mid t \in \mathcal{S}\}.$$

A *substitution* σ is a mapping from $\mathcal{V}$ into $\mathcal{T}(\Sigma, \mathcal{V})$ where only finitely many variables are *not* mapped to themselves. If t is a term, then $\sigma(t)$ shall denote the term obtained from t by simultaneously replacing every variable v occurring in t by $\sigma(v)$.

Given a ground term t and a term t', we say that t' and t *match* if there exists a substitution σ such that $\sigma(t') = t$; in this case σ is called a *matcher* for t and t'.

3 Tree Transducers

Tree transducers [12] perform transformations on terms. They obtain a term as input and return (another) term as output. Non-deterministic tree transducers may produce different output terms on one input term. Here, we consider (non-deterministic) top-down tree transducers with regular look-ahead [10], which work as follows: They read the head symbol f of the input term $f(t_0, \ldots, t_{n-1})$, depending on the current state replace it by some term, and then proceed to transform the subterms t_i in the same fashion. Before the transformation takes place, the transducer may check whether the subterms t_i belong to certain regular tree languages. In what follows, we first introduce top-down tree transducers without look-ahead and then those with look-ahead.

Formally, a *top-down tree transducer* (TT) $\mathcal{A}$ is a tuple $(Q, \Sigma, \Sigma', I, \Delta)$ where Q is a finite set of *states*, considered as unary symbols, Σ and Σ' are ranked alphabets, the input and output alphabet, respectively, $I \subseteq Q$ is the set of *initial states*, and Δ is a finite set of *transitions* of the form

$$q(f(v_0, \ldots, v_{n-1})) \to C[q_0(v_{i_0}), \ldots, q_{l-1}(v_{i_{l-1}})] \tag{1}$$

where $l \geq 0$, $q, q_0, \ldots, q_{l-1} \in Q$, f is an n-ary symbol, $v_0, \ldots, v_{n-1}$ are distinct variables, $C \in \mathcal{C}^l(\Sigma')$ is an l-ary context, and $i_0, \ldots, i_{l-1} < n$ are not necessarily distinct indices. If $q \in Q$, then $\mathcal{A}_q$ denotes the TT $\mathcal{A}$ where q is the only initial state.

The computation of a TT $\mathcal{A}$ is defined as follows. If $t, t' \in \mathcal{T}(\Sigma \cup \Sigma' \cup Q)$, then the *one step relation* $\to_{\mathcal{A}}$ of $\mathcal{A}$ is given by: $t \to_{\mathcal{A}} t'$ iff there exists

- $q(f(v_0, \ldots, v_{n-1}) \to C[q_0(v_{i_0}), \ldots, q_{l-1}(v_{i_{l-1}})] \in \Delta$,
- a unary context $C' \in \mathcal{C}^1(\Sigma \cup \Sigma' \cup Q)$, and
- ground terms $t_0, \ldots, t_{n-1} \in \mathcal{T}(\Sigma)$

such that $t = C'[q(f(t_0, \ldots, t_{n-1}))]$ and $t' = C'[C[q_0(t_{i_0}), \ldots, q_{l-1}(t_{i_{l-1}})]]$.

By $\to^*_{\mathcal{A}}$ we denote the *computation relation* of $\mathcal{A}$, i.e., the reflexive and transitive closure of $\to_{\mathcal{A}}$. A ground term $t \in \mathcal{T}(\Sigma)$ is *transformed* by $\mathcal{A}$ to a ground term $t' \in \mathcal{T}(\Sigma')$ if $q(t) \to^*_{\mathcal{A}} t'$ for some $q \in I$. The *transformation* induced by $\mathcal{A}$ is the binary relation

$$\mathcal{L}(\mathcal{A}) := \{(t, t') \in \mathcal{T}(\Sigma) \times \mathcal{T}(\Sigma') \mid q(t) \to^*_{\mathcal{A}} t' \text{ for some } q \in I\}.$$

We note that if in a TT $\mathcal{A}$ all transitions are of the form

$$q(f(v_0, \ldots, v_{n-1})) \to f(q_0(v_0), \ldots, q_{n-1}(v_{n-1})),$$

then $\mathcal{A}$ can be interpreted as a *top-down tree automaton* (TDTA). $\mathcal{A}$ *accepts* a term t iff $q(t) \to^*_{\mathcal{A}} t$ for some $q \in I$. The set of terms accepted by $\mathcal{A}$ is also denoted by $\mathcal{L}(\mathcal{A})$.

A *top-down tree automaton with regular look-ahead* (TTLA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \Sigma', I, \Delta, \mathcal{G})$ where Q, Σ, Σ', and I are defined just as for TTs, $\mathcal{G}$ is a top-down tree automaton, the *guard automaton*, and Δ is a finite set of transitions of the form

$$q(f(v_0, \ldots, v_{n-1})) \overset{p_0,\ldots,p_{n-1}}{\longrightarrow} C[q_0(v_{i_0}), \ldots, q_{l-1}(v_{i_{l-1}})] \tag{2}$$

where $p_0, \ldots, p_{n-1}$ are (not necessarily distinct) states of $\mathcal{G}$, the *guard of the transition*, and everything else is defined as in (1). Sometimes we omit the guard of a transition. This is an abbreviation for the guard $p, \ldots, p$ with $\mathcal{L}(\mathcal{G}_p) = \mathcal{T}(\Sigma)$, i.e., omitting the guard means that there are no restrictions on the application of the transition. We may assume that $\mathcal{G}$ always contains such a state p.

The one step relation $\rightarrow_{\mathcal{A}}$ of $\mathcal{A}$ is defined analogously to the one step relation for TTs. The only difference is that a transition of the form (2) can only be applied if $t_i \in \mathcal{L}(\mathcal{G}_{p_i})$ for every $i < n$, where t_i is the term substituted for v_i. Again, the computation relation $\rightarrow^*_{\mathcal{A}}$ is the reflexive and transitive closure of the one step relation $\rightarrow_{\mathcal{A}}$.

We now define another extension of TTs where the left hand-side of a transition is some linear term and show that these transducers are not more powerful than TTLAs. Using these transducers, it is easy to see that they can simulate all rewrite rules $t \rightarrow t'$, as used in models for non-looping protocols, in case t is linear. Thus, TTLAs can simulate these rules as well.

Formally, a *TT with linear left hand-side (TTLL)* $\mathcal{A}$ is a TT with transitions of the form:

$$q(t(v_0, \ldots, v_{n-1})) \rightarrow C[q_0(v_{i_0}), \ldots, q_{l-1}(v_{i_{l-1}})] \tag{3}$$

where everything is defined as in (1), but $t(v_0, \ldots, v_{n-1})$ is some linear term containing the variables $v_0, \ldots, v_{n-1}$. We assume that $t(v_0, \ldots, v_{n-1})$ is not a variable. The one step relation and the computation relation induced by $\mathcal{A}$ are defined in the obvious way.

The following lemma states that every TTLL can be turned into an equivalent TTLA.

Lemma 1 *For every TTLL $\mathcal{A} = (Q, \Sigma, \Sigma', I, \Delta)$ there exists a TTLA $\mathcal{B} = (Q', \Sigma, \Sigma', I, \Delta', \mathcal{G})$ of size polynomially bounded in the size of $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}_q) = \mathcal{L}(\mathcal{B}_q)$ for every $q \in Q$.*

Proof. The idea of the construction of $\mathcal{B}$ is as follows: We use the guards to check whether a messages matches $t(v_0, \ldots, v_{n-1})$ in (3). To reach a term substituted for some v_i we introduce transitions that navigate to the corresponding position.

Formally, let $\mathcal{S}$ denote the set of subterms occurring on the left hand-side of transitions of $\mathcal{A}$. The guard automaton $\mathcal{G}$ of $\mathcal{B}$ is constructed in such a way that $\mathcal{G}$ contains for every (linear) term $t(v_0, \ldots, v_{n-1}) \in \mathcal{S} \cap \mathcal{T}(\Sigma, \mathcal{V})$ a state q_t such that $\mathcal{L}(\mathcal{G}_{q_t}) = \{t[t_0, \ldots, t_{n-1}] \in \mathcal{T}(\Sigma) \mid t_i \in \mathcal{T}(\Sigma), i < n\}$. It is easy to define such an automaton with size polynomially bounded in the size of $\mathcal{A}$.

Let $\mathcal{P} \subseteq \omega^*$ be the (finite) set of positions of terms in $\mathcal{S}$. We define $Q' := Q \cup \{p_{\pi,q} \mid \pi \in \mathcal{P}, q \in Q\}\}$. A state of the form $p_{\varepsilon,q}$ is identified with q. For every n-ary symbol $f \in \Sigma$, $q \in Q$, $i \geq 0$, and π with $i\pi \in \mathcal{P}$, $\mathcal{B}$ contains the following transition:

$$p_{i\pi,q}(f(v_0, \ldots, v_{n-1})) \to p_{\pi,q}(v_i),$$

where $v_0, \ldots, v_{n-1}$ are some distinct variables. Recall that $p_{\varepsilon,q} = q$.

For every transition of the form (3) in $\mathcal{A}$ with $t(v_0, \ldots, v_{n-1})$ of the form

$$f(t_0(v_0, \ldots, v_{n-1}), \ldots, t_{n'-1}(v_0, \ldots, v_{n-1})),$$

$\mathcal{B}$ contains the transition

$$q(f(v'_0, \ldots, v'_{n'-1})) \stackrel{p_{t_0}, \ldots, p_{t_{n'-1}}}{\longrightarrow} C[p_{\pi_{v_{i_0}},q_0}(v'_{i'_0}), \ldots, p_{\pi_{v_{i_{l-1}}},q_{l-1}}(v'_{i'_{l-1}})]$$

where i'_j is the uniquely determined index such that v_{i_j} occurs in $t_{i'_j}$ and $\pi_{v_{i_j}}$ denotes the position of v_{i_j} in $t_{i'_j}$.

One easily checks that $\mathcal{B}$ meets the required conditions. $\square$

As explained in the introduction, the transformation induced by a rewrite rule $t \to t'$ with $t \in \mathcal{T}(\Sigma, \{v_0, \ldots, v_{n-1}\})$ and $t' \in \mathcal{T}(\Sigma', \{v_0, \ldots, v_{n-1}\})$ is

$$\begin{aligned}\mathcal{L}(t \to t') := \{(s, s') \in \mathcal{T}(\Sigma) \times \mathcal{T}(\Sigma') \mid & \text{there exists a substitution } \sigma \\ & \text{such that } s = \sigma(t) \text{ and } s' = \sigma(t')\}.\end{aligned}$$

Using TTLLs, it is now easy to see that every transformation induced by a rewrite rule with linear left hand-side can be simulated by a TTLL (and thus, by a TTLA).

Lemma 2 *For every rewrite rule $t(v_0, \ldots, v_{n-1}) \to t'(v_0, \ldots, v_{n-1})$ with $Var(t') \subseteq Var(t)$, t linear, and not a variable, there exists a TTLL (TTLA) $\mathcal{A}$ such that $\mathcal{L}(t \to t') = \mathcal{L}(\mathcal{A})$.*

Proof. Due to Lemma 1, it suffices to show that there exists a TTLL $\mathcal{A}$ equivalent to the given rewrite rule $t \to t'$. We define the following TTLL $\mathcal{A}$: The set of states consists of q_I and q_{copy} where q_I is the initial state. The transitions are:

$$\begin{aligned} q_I(t(v_0, \ldots, v_{n-1})) &\to t'[q_{copy}(v_0), \ldots, q_{copy}(v_{n-1})], \\ q_{copy}(f(v_0, \ldots, v_{n-1})) &\to f(q_{copy}(v_0), \ldots, q_{copy}(v_{n-1})) \end{aligned}$$

for every n-ary symbol $f \in \Sigma$. That is, in state q_I $\mathcal{A}$ performs the transformation according to the rule $t \to t'$ and in state q_{copy} subterms are copied. Obviously, $\mathcal{L}(t \to t') = \mathcal{L}(\mathcal{A})$. $\square$

4 The Tree Transducer-based Protocol Model

The assumptions underlying our protocol model basically coincide with those for models of non-looping protocols in which security is decidable (see, for example, [23, 1, 19]): The intruder is modeled by the standard Dolev-Yao intruder where no restrictions are put on the size of messages; principals and the intruder cannot generate nonces; keys are assumed to be atomic messages, i.e., constants, rather than arbitrary messages (in some of the models cited above they may be arbitrary messages); the number of sessions considered in the analysis of a protocol is bounded — more precisely, the sessions considered are only those encoded in the description of the protocol itself.

The main difference between our model and other models is that receive-send actions are described by tree transducers rather than by single rewrite rules that are applied only once to the input message (see the introduction).

We now give the formal definition of our tree transducer-based protocol model by defining messages, the intruder, protocols, and attacks.

4.1 Messages

The definition of messages is rather standard. Let $\mathcal{N}$ denote a finite set of *atomic messages*, containing keys, names of principals, etc. as well as the special atomic message secret. Let $\Sigma_{\mathcal{N}}$ denote the signature consisting of the constants $\mathcal{N}$, the unary symbols $\mathtt{hash}$ and $\mathtt{enc}_a$ for every $a \in \mathcal{N}$, and the binary symbol $<>$. Instead of $<>(t, t')$ we write $<t, t'>$. The *set of messages* $\mathcal{M}$ (over $\mathcal{N}$) is the set $\mathcal{T}(\Sigma_{\mathcal{N}})$ of ground terms over $\Sigma_{\mathcal{N}}$. Message are usually denoted by m, x, or y or decorations thereof.

4.2 The Intruder

We use the standard Dolev-Yao intruder model [9]. 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 $\mathsf{d}(\mathcal{K})$ the intruder can derive from $\mathcal{K} \subseteq \mathcal{M}$ is the smallest set satisfying the following conditions:

- $\mathcal{K} \subseteq \mathsf{d}(\mathcal{K})$;
- if $<m, m'> \in \mathsf{d}(\mathcal{K})$, then $m, m' \in \mathsf{d}(\mathcal{K})$ (decomposition);
- if $\mathtt{enc}_a(m) \in \mathsf{d}(\mathcal{K})$ and $a \in \mathsf{d}(\mathcal{K})$, then $m \in \mathsf{d}(\mathcal{K})$ (decryption);
- if $m, m' \in \mathsf{d}(\mathcal{K})$, then $<m, m'> \in \mathsf{d}(\mathcal{K})$ (composition);
- if $m \in \mathsf{d}(\mathcal{K})$ and $a \in \mathcal{N} \cap \mathsf{d}(\mathcal{K})$, then $\mathtt{enc}_a(m) \in \mathsf{d}(\mathcal{K})$ (encryption);
- if $m \in \mathsf{d}(\mathcal{K})$, then $\mathtt{hash}(m) \in \mathsf{d}(\mathcal{K})$ (hashing).

Let $\mathsf{an}(\mathcal{K})$ denote the closure of $\mathcal{K}$ under decomposition and decryption, and $\mathsf{syn}(\mathcal{K})$ the closure of $\mathcal{K}$ under composition, encryption, and hashing. Note that every message in $\mathsf{an}(\mathcal{K})$ is a subterm of some message in $\mathcal{K}$.

It is well-known that, since we only allow for atomic keys, $\mathsf{d}(\mathcal{K})$ can be obtained by first taking the closure of $\mathcal{K}$ under decomposition and decryption, and from this the closure under composition, encryption, and hashing (see, e.g., [21]):

Lemma 3 *For every $\mathcal{K} \subseteq \mathcal{M}$: $\mathsf{d}(\mathcal{K}) = \mathsf{syn}(\mathsf{an}(\mathcal{K}))$.*

4.3 Protocols

Protocols are described by sets of principals and every principal is defined by a sequence of receive-send actions, which in a protocol run are performed one after the other. Every receive-send action is specified by a certain tree transducer, which we will call message transducer.

Definition 4 *A* message transducer $\mathcal{A}$ *is a tuple* $(Q, \mathcal{N}, I, \Delta, \mathcal{G})$ *such that* $(Q, \Sigma_{\mathcal{N}}, \Sigma_{\mathcal{N}}, I, \Delta, \mathcal{G})$ *is a TTLA, where* $\mathcal{N}$ *and* $\Sigma_{\mathcal{N}}$ *are defined as above.*

Roughly speaking, a principal is defined as a sequence of message transducers.

Definition 5 *A* (TTLA-based) principal Π *is a tuple* $((A_0, \ldots, A_{n-1}), I)$ *consisting of a sequence* $(A_0, \ldots, A_{n-1})$ *of message transducers* $\mathcal{A}_i = (Q_i, \mathcal{N}, I_i, \Delta_i, \mathcal{G}_i)$ *and an* n*-ary relation* $I \subseteq I_0 \times \cdots \times I_{n-1}$.

The single message transducers $\mathcal{A}_i$ in the definition of Π are called *receive-send actions*. In a protocol run, Π performs the receive-send actions one after the other. More precisely, at the beginning of a protocol run, a tuple $(q_0, \ldots, q_{n-1}) \in I$ is chosen non-deterministically where q_i will be the initial state of $\mathcal{A}_i$. Now, if in the protocol run the first message Π receives is m_0, then Π returns some message m'_0 with $q_0(m_0) \to^*_{\mathcal{A}_0} m'_0$. Then, on receiving the second message, say m_1, Π returns m'_1 with $q_1(m_1) \to^*_{\mathcal{A}_1} m'_1$, and so on. By fixing the initial states at the beginning, we model that Π can convey (a finite amount of) information from one receive-send action to another. For example, if q_0 encodes that Π expects to talk to Bob, then q_1 might describe that in the second message Π expects to see Bob's name again.

We note that instead of a sequence of receive-send actions we could as well describe principals by a partial ordering of receive-send actions (as in [23]) without any impact on the decidability and complexity theoretic result.

A protocol is defined as a finite family of principals plus, since we are interested in attacks on this protocol, the initial intruder knowledge.

Definition 6 *A* (TTLA-based) protocol P *is a tuple* $(\{\Pi_i\}_{i<n}, \mathcal{K})$ *where*

- $\{\Pi_i\}_{i<n}$ *is a family of* n *(TTLA-based) principals, and*
- $\mathcal{K} \subseteq \mathcal{M}$ *is a finite set, the* initial intruder knowledge.

4.4 Attacks on Protocols

In an attack on a protocol, the receive-send actions of the principals are interleaved in some way and the intruder, who has complete control over the communication, tries to produce inputs for the principals such that from the corresponding outputs and his initial knowledge he can derive the secret message secret. Formally, an attack is defined as follows.

Definition 7 *Let $P = (\{\Pi_i\}_{i<n}, \mathcal{K})$ be a protocol with $\Pi_i = ((\mathcal{A}_0^i, \ldots, \mathcal{A}_{n_i-1}^i), I_i)$ and $I_i \subseteq I_0^i \times \cdots \times I_{n_i-1}^i$ for $i < n$. An* attack *on P is a tuple consisting of the following components:*

- *a total ordering $<$ on the set $\{(i,j) \mid i < n, j < n_i\}$ such that $(i,j) < (i,j')$ implies $j < j'$ (the execution order of the receive-send actions);*
- *a mapping ψ assigning to every (i,j), $i < n$, $j < n_i$, a tuple $\psi(i,j) = (q_j^i, m_j^i, {m'}_j^i)$*

such that for every $i < n$ and $j < n_i$:

- $(q_0^i, \ldots, q_{n_i-1}^i) \in I_i$,
- $m_j^i, {m'}_j^i \in \mathcal{M}$ *and* $q_j^i(m_j^i) \rightarrow^*_{\mathcal{A}_j^i} {m'}_j^i$, *and*
- $m_j^i \in \mathsf{d}(\mathcal{K} \cup \{{m'}_{j'}^{i'} \mid (i',j') < (i,j)\})$.

An attack is called successful *if* $\mathsf{secret} \in \mathsf{d}(\mathcal{K} \cup \{{m'}_j^i \mid i < n, j < n_i\})$.

The decision problem we are interested in is the following:

Attack: Given a protocol P, decide whether there exists a successful attack on P.

A protocol guarantees *secrecy* if there does not exist a successful attack. In this case, we say that the protocol is *secure*.

5 Undecidability Results

We show that different extensions of the TTLAs lead to undecidability of Attack. This motivates the kind of TTLA we consider here and demonstrates the limitations of automatic analysis of iterative protocols. More precisely, we show that in the following cases Attack is undecidable.

- When we allow TTLAs where the terms on the left hand-side of transitions may be non-linear, i.e., in (2) the variables v_i are not necessarily distinct — in this case, a principal can test arbitrary large messages for equality.
- When TTLAs can store one message of arbitrary size and compare it with the input message for equality.
- When we allow TTLAs where in the right hand-side of transitions the state symbols can occur everywhere in the term instead of only at variable positions.

Finally, we will show that when an intruder is allowed to use an unbounded number of copies of a principal to perform an attack, i.e., the protocol is analyzed w.r.t. an unbounded number of sessions, then ATTACK is also undecidable. This is not surprising. The same is true for models of non-looping protocols (see, e.g., [1]).

These undecidability results are shown in the following subsections.

5.1 TTLAs with Non-linear Left Hand-side

We define *top-down tree transducers with non-linear left hand-side* (TTNL) to be TTLAs with transitions of the form as defined in (2) but where the variables $v_0, \ldots, v_{n-1}$ are not required to be distinct. A protocol where the receive-send actions are defined by TTNLs is called a *TTNL-based protocol*. Alternatively, to establish the undecidability result, TTNLs could also be defined by TTLLs with transitions of the form as specified in (3) but where $t(v_0, \ldots, v_{n-1})$ may be non-linear.

We show the following:

Theorem 8 *For TTNL-based protocols,* ATTACK *is undecidable.*

The proof of the theorem is by reduction from Post's Correspondence Problem (PCP).

An instance of PCP is defined as follows: Given an alphabet Σ with at least two letters and two sequences $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ of words over the alphabet Σ (including the empty word ε), decide whether there exist indices $i_0, \ldots, i_{k-1}$, $k > 0$, such that $\alpha_{i_0} \ldots \alpha_{i_{k-1}} = \beta_{i_0} \ldots \beta_{i_{k-1}}$.

We will encode a word $\alpha = a_0 \ldots a_{l-1} \in \Sigma^*$ with $a_i \in \Sigma$ by the term $t_\alpha := <a_0, <a_1, \ldots, <a_{l-1}, \perp> \cdots >$, where $\perp$ is a new constant. We also have to encode the indices $i \in \{1, \ldots, n\}$. Let b be a new constant and let b^i denote the word $b \cdots b$ of length i. Then, i is encoded by $t_i = t_{b^i}$. In addition to b, we will use the new constants b' and b''. The set of atomic messages is defined to be $\mathcal{N} := \Sigma \cup \{b, b', b'', \perp, \mathsf{secret}\}$.

The idea of the reduction is as follows: We use one principal who performs three receive-send actions. The initial intruder knowledge is $\Sigma \cup \{b, \perp\}$. The intruder guesses a sequence $i_0, \ldots, i_{k-1}$ of indices encoded as $m_0 = <t_{i_0}, <t_{i_1}, \ldots, <t_{i_{k-1}}, \perp> \cdots >$ and sends m_0 to the principal. In his first receive-send action the principal duplicates m_0 and encrypts it with b', i.e., the principal returns $m_0' = \mathsf{enc}_{b'}(<m_0, m_0>)$. The encryption is done to prevent the intruder from changing this message. In the second receive-send action the principal will only accept messages encrypted by b'. Thus, the intruder must send m_0' to the principal. Now, the principal performs the second receive-send action by reading m_0' and turning the m_0's into words over Σ (encoded as terms) by replacing every index i by the corresponding word α_i, for the left m_0, and β_i, for the right m_0. The message, m_1', the second receive-send action returns is encrypted by b'', for the same reason the first message was encrypted by b'. Now, in the last receive-send action the principal reads m_1' and checks whether the two words encoded in m_1' are equal. This can be done because the left hand-sides of transitions may be non-linear. If the two words coincide, the secret message secret is given to the intruder. It is easy to see that the intruder only obtains the secret if he was able to guess a solution of the instance of the PCP.

More formally, the TTNLs, call them $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$, describing the three receive-send actions informally explained above are defined as follows. The state q_I will always denote the initial state. The TTNL $\mathcal{A}_0$ is given by one transition:

$$q_I(<v,v'>) \to \mathtt{enc}_{b'}(<<v,v'>,<v,v'>>).$$

By writing $<v,v'>$ we make sure that the sequence of indices is not empty. The transitions of $\mathcal{A}_1$ are:

$$q_I(\mathtt{enc}_{b'}(<v,v'>)) \to \mathtt{enc}_{b''}(<q_\alpha(v),q_\beta(v')>)$$

and for every $i \in \{1,\ldots,n\}$ with $\alpha_i = a_0 \cdots a_{l-1}$:

$$\begin{array}{rcl} q_\alpha(<t_i,v>) & \to & <a_0,<a_1,\ldots,<a_{l-1},q_\alpha(v)>\cdots> \\ q_\alpha(\perp) & \to & \perp. \end{array}$$

The transitions for the β_i's are defined analogously. Finally, $\mathcal{A}_2$ is given by two transitions:

$$\begin{array}{rcl} q_I(\mathtt{enc}_{b''}(v)) & \to & q_=(v), \\ q_=(<v,v>) & \to & \mathsf{secret}. \end{array}$$

One easily shows that the instance of the PCP has a solution iff there exists a successful attack on the protocol just described. This shows Theorem 8. We point out that the $\mathcal{A}_i$'s used in the reduction are deterministic. Also, instead of considering one principal performing three receive-send actions one could have defined a protocol with three principals each performing one receive-send action.

5.2 One Memory TTs

We define *one memory top-down tree transducers* (OMTTs) as follows: The *generalized state* of an OMTT is a tuple (q,m) where q is an element of a finite state space and m is a message or $\perp$. Just as for TTLAs, the transitions of an OMTT are of the form (2) where the states are replaced by generalized states of the form $(q,\perp)$ — alternatively, one could consider transitions of TTLLs. In addition, an OMTT can have a *copy transition*

$$(q,\perp)(<x,y>) \to (q_=,x)(y)$$

where q is some element of the finite state space and $q_=$ is a fixed state, the *equality test state*. This copy transition allows the transducer to write a term into its memory. In state $q_=$ the transducer then compares the memory with the input term. Formally, for every l-ary symbol f, distinct variables $v_0,\ldots,v_{l-1},v'_0,\ldots,v'_{l-1}$, $l \geq 1$, and constant a, an OMTT may contain the *equality test transitions*:

$$(q_=,a)(a) \to \mathsf{secret}, \text{ and}$$

$$(q_=, f(v_0, \ldots, v_{l-1}))(f(v'_0, \ldots, v'_{l-1}))$$
$$\downarrow$$
$$<(q_=, v_0)(v'_0), <\ldots, <(q_=, v_{l-2})(v'_{l-2}), (q_=, v_{l-1})(v'_{l-1})> \cdots >$$

The computation of an OMTT is defined in the obvious way. Protocols where the receive-send actions are defined by OMTTs are called *OMTT-based protocols*.

Clearly, with an OMTT one can describe the receive-send actions $\mathcal{A}_0$, $\mathcal{A}_1$, and the first transition of $\mathcal{A}_2$ from Section 5.1. The last transition of $\mathcal{A}_2$ can be simulated by first applying the copy transition and then the equality test transition. The result of the equality test is, if it succeeds, a term consisting of pairing symbols and secret. Thus, the intruder will be able to derive secret. This shows:

Theorem 9 *For OMTT-based protocols,* ATTACK *is undecidable.*

5.3 TTs with Generalized Right Hand-side

A *top-down tree transducer with generalized right hand-side* (TTR) is defined as a TTLA with transitions of the form

$$q(f(v_0, \ldots, v_{n-1})) \overset{p_0, \ldots, p_{n-1}}{\longrightarrow} C[v_{i_0}, \ldots, v_{i_{l-1}}]$$

where C is some l-ary context over $\Sigma_{\mathcal{N}} \cup Q$. Thus, the difference to transitions of the form (2) is that in C the state symbols need not be attached to the variables v_i. Now, *TTR-based protocols* are defined as expected. In the same way one can generalize transitions of TTLLs.

With the following transitions, equality tests can be performed:

$$q_=(<a, a>) \quad \to \quad \mathsf{secret}, \text{ and}$$

$$q_=(<f(v_0, \ldots, v_{l-1}), f(v'_0, \ldots, v'_{l-1})>)$$
$$\downarrow$$
$$<q_=(<v_0, v'_0>), <\cdots <q_=(<v_{l-2}, v'_{l-2}>), q_=(<v_{l-1}, v'_{l-1}>)> \cdots >$$

These are generalized TTLL-transitions. However, using a similar construction as in the proof of Lemma 1, they can easily be turned into TTR-transitions.

As in the previous section we conclude:

Theorem 10 *For TTR-based protocols,* ATTACK *is undecidable.*

5.4 Unbounded Number of Sessions

So far, in an attack on a (TTLA-based) protocol $P = (\{\Pi_i\}_{i<n}, \mathcal{K})$ the intruder can use every Π_i only once. We now generalize the notion of attack in that the intruder can use an unbounded number of copies of every Π_i to conduct his attack. Formally, there exists a *successful* ∞*-attack* on P if there exists a successful attack on some protocol $P_\infty = (M, \mathcal{K})$ where M is a family of principals of the form $\{\Pi_0^0, \ldots, \Pi_0^{l_0-1}, \ldots, \Pi_{n-1}^0, \ldots, \Pi_{n-1}^{l_{n-1}-1}\}$ and every Π_i^j is a copy of Π_i. In other words, in an ∞-attack on P, the intruder may use l_i instances of Π_i, for any l_i, to perform

an attack on P. We call ∞-ATTACK the problem of deciding whether there exists a successful ∞-attack on a (TTLA-based) protocol.

To see that ∞-ATTACK is undecidable, consider the protocol P that consist of two principals. One principal performs two receive-send actions described by $\mathcal{A}_0$ and $\mathcal{A}_1$ as defined in Section 5.1 and the other principal performs one receive-send action $\mathcal{A}$ defined by the following transitions where $a \in \Sigma$:

$$
\begin{aligned}
q_I(\mathsf{enc}_{b''}(<<a,v>,<a,v'>>)) &\rightarrow \mathsf{enc}_{b''}(<v,v'>), \\
q_I(\mathsf{enc}_{b''}(<\perp,\perp>)) &\rightarrow \mathsf{secret}.
\end{aligned}
$$

It is easy to see that the instance of the PCP as defined in Section 5.1 has a solution if and only if there exists an ∞-attack on P. This shows:

Theorem 11 *For TTLA-based protocols,* ∞-ATTACK *is undecidable.*

6 The Decidability Result

We show the following theorem.

Theorem 12 *For TTLA-based protocols,* ATTACK *is decidable.*

To prove this theorem it obviously suffices to show that the following problem is decidable.

MESSAGEPROBLEM. *Given a finite set $\mathcal{K} \subseteq \mathcal{M}$ and $k \geq 0$ message transducers $\mathcal{A}_0, \ldots, \mathcal{A}_{k-1}$ with $\mathcal{A}_i = (Q_i, \mathcal{N}, \{q_i^I\}, \Delta_i, \mathcal{G}_i)$ for $i < k$, decide whether there exist messages $m_i, m_i' \in \mathcal{M}$, $i < k$, such that*

1. $m_i \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{i-1}'\})$ *for every* $i < k$,
2. $q_i^I(m_i) \rightarrow^*_{\mathcal{A}_i} m_i'$ *for every* $i < k$, *and*
3. $\mathsf{secret} \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{k-1}'\})$.

For simplicity of notation and w.l.o.g. we assume that all guard automata $\mathcal{G}_i$ coincide. In what follows, we denote this guard automaton by $\mathcal{G}$. We write an instance of the MESSAGEPROBLEM as $(\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$ and a solution of such an instance as a tuple $(m_0, m_0', \ldots, m_{k-1}, m_{k-1}')$ of messages.

6.1 Deciding the Message Problem

To show that MESSAGEPROBLEM is decidable, we first show for every $i < k$:

(*) If $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ is a solvable instance of MESSAGEPROBLEM, then there exists a solution $(m_i, m_i', \ldots, m_{k-1}, m_{k-1}')$ with $\mathsf{depth}(m_i)$ bounded by an (effectively computable) function in the size of the instance.

Given (*), the proof proceeds as follows. We observe that the depth of an output term of a TTLA can be bounded by some (effectively computable) function in the depth of the input term. Roughly speaking, the reason is that in every computation step of the TTLA a subterm of the input message is replaced by a bounded number of subterms of smaller depth. Using this idea, it is easy to show:

Lemma 13 *For every TTLA $\mathcal{A}$ there exists an effectively computable function* $\mathsf{depth}_{\mathcal{A}} : \mathbf{N} \to \mathbf{N}$, *where* $\mathbf{N}$ *is the set of non-negative integers, such that for every state q of $\mathcal{A}$ and ground terms t, t' with $q(t) \to_{\mathcal{A}}^{*} t'$: $|t'| \leq \mathsf{depth}_{\mathcal{A}}(|t|)$.*

By induction on i, and using (*) and Lemma 13, we obtain:

Lemma 14 *There exists an effectively computable function* $\mathsf{depth}_{MP} : \mathbf{N} \to \mathbf{N}$ *such that for every solvable instance $I = (\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$ of* MESSAGEPROBLEM *there exists a solution $(m_0, m'_0, \ldots, m_{k-1}, m'_{k-1})$ with $\mathsf{depth}(m_i) \leq \mathsf{depth}_{MP}(|I|)$ and $\mathsf{depth}(m'_i) \leq \mathsf{depth}_{MP}(|I|)$ for every $i < k$.*

Now, given an instance I of MESSAGEPROBLEM, an algorithm can guess a solution $(m_0, m'_0, \ldots, m_{k-1}, m'_{k-1})$ with the depth of the messages bounded by $\mathsf{depth}_{MP}(|I|)$ and check whether all conditions 1. to 3. are met (which obviously can be decided). As a consequence, we obtain:

Proposition 15 MESSAGEPROBLEM *is decidable.*

From this, Theorem 12 follows immediately.

In what follows, the proof for (*), which is stated more precisely in Proposition 21 below, is given.

6.2 Bounding the Depth of m_i

We will apply a pumping argument to bound the depth of m_i. To this end, first different quasi-orderings, i.e., reflexive and transitive orderings, are introduced.

For messages m and m' we define $m \preceq_{\mathcal{G}} m'$ iff for every state q in $\mathcal{G}$: $m \in \mathcal{L}(\mathcal{G}_q)$ implies $m' \in \mathcal{L}(\mathcal{G}_q)$.

For the following quasi-ordering we need some notation. For messages $m_0, \ldots,$ m_{n-1}, and $\mathcal{K}, \mathcal{K}' \subseteq \mathcal{M}$ we write $\mathsf{an}(m_0, \ldots, m_{n-1}, \mathcal{K})$ instead of $\mathsf{an}(\{m_0, \ldots, m_{n-1}\} \cup \mathcal{K})$ and $\mathsf{an}_{\mathcal{K}'}(m_0, \ldots, m_{n-1}, \mathcal{K})$ instead of $\mathsf{an}(\{m_0, \ldots, m_{n-1}\} \cup \mathcal{K}) \cap \mathcal{K}'$.

Definition 16 *The* atom preserving ordering $\preceq$ *is defined as follows. For messages $m, m' \in \mathcal{M}$: $m \preceq m'$ iff for all $N \subseteq \mathcal{N}$: $\mathsf{an}_{\mathcal{N}}(m, N) \subseteq \mathsf{an}_{\mathcal{N}}(m', N)$.*

Intuitively, $m \preceq m'$ says that in every context the set of atomic messages derivable from m is a subset of the atomic messages derivable from m'. For example, for $a, a', a'' \in \mathcal{N}$, $\mathsf{enc}_a(\mathsf{enc}_{a'}(a'')) \preceq \langle a', \mathsf{enc}_{a'}(\mathsf{enc}_a(a''))\rangle$.

The most important orderings are the solvability preserving ordering $\preceq_i$ and the computation preserving ordering $\sqsubseteq_i$, $i \leq k$. These orderings depend on the message transducers $\mathcal{A}_i, \ldots, \mathcal{A}_{k-1}$ and are defined simultaneously by induction. The idea behind these orderings is as follows: We want to guarantee that if $m \preceq_i m'$ and $(\mathcal{K}, \mathcal{A}_i, \ldots, A_{k-1})$ has a solution, say $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$, then so has $(\mathcal{K} \setminus \{m\} \cup \{m'\}, \mathcal{A}_i, \ldots, A_{k-1})$. In other words, the ordering $\preceq_i$ is solvability preserving. The message m_i may have been built up from submessages x of m. Now, if m is replaced by m', we need to make sure that in m' there is a submessage x' that can be used instead of x. This is why we need Condition 3. in the definition below. The relation

$m \preceq m'$ in this definition ensures that when replacing m by m' we have the same set of atomic messages (or a superset) to construct new messages. Condition 1. makes sure that $\preceq_i$ refines $\preceq_{i+1}$. The computation preserving ordering $m \sqsubseteq_i m'$ says that the transformation $\mathcal{A}_i$ performs on m can also be carried out on m'. Condition 2. in the definition of $\preceq_i$ is needed to guarantee that $\preceq_i$ is closed under substitution.

Definition 17 *For every $i \leq k$, the* solvability preserving ordering $\preceq_i$ *is defined as follows: for all $m, m' \in \mathcal{M}$, $m \preceq_i m'$ iff $m \preceq m'$ and if $i < k$, then*

1. *$m \preceq_{i+1} m'$,*
2. *$m \sqsubseteq_i m'$, and*
3. *for every $N \subseteq \mathcal{N}$ and $x \in \mathsf{an}(m, N)$, there exists $x' \in \mathsf{an}(m', N)$ such that $x \sqsubseteq_i x'$.*

For $i < k$ the computation preserving ordering $\sqsubseteq_i$ *is defined as follows: For messages $m, m' \in \mathcal{M}$, we define $m \sqsubseteq_i m'$ iff $m \preceq_{\mathcal{G}} m'$, and for every $q \in Q_i$ and $y \in \mathcal{M}$, $q(m) \rightarrow^*_{\mathcal{A}_i} y$ implies that there exists $y' \in \mathcal{M}$ with $q(m') \rightarrow^*_{\mathcal{A}_i} y$ and $y \preceq_{i+1} y'$.*

To show that $\preceq_i$ is solvability preserving, we first need to show that $\preceq_i$ and $\sqsubseteq_i$ are closed under substitution (see Appendix A for the proof).

Lemma 18 *Let $x_0, \ldots, x_{n-1}, x'_0, \ldots, x'_{n-1} \in \mathcal{M}$, $i \leq k$, and C be an n-ary context. If $x_j \preceq_i x'_j$ for all $j < n$, then $C[x_0, \ldots, x_{n-1}] \preceq_i C[x'_0, \ldots, x'_{n-1}]$. If $i < k$ and $x_j \sqsubseteq_i x'_j$ for all $j < n$, then $C[x_0, \ldots, x_{n-1}] \sqsubseteq_i C[x'_0, \ldots, x'_{n-1}]$.*

Now, we can conclude (see Appendix B for the proof):

Proposition 19 *Assume that $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$, $i \leq k$, is a solvable instance of* MESSAGEPROBLEM *and $m \in \mathcal{K}$. Then, for every $\overline{m} \in \mathcal{M}$ with $m \preceq_i \overline{m}$, the instance $(\overline{\mathcal{K}}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ with $\overline{\mathcal{K}} := \mathcal{K} \setminus \{m\} \cup \{\overline{m}\}$ is also solvable.*

Let $=_i$ and $\equiv_i$ denote the equivalence relations induced by $\sqsubseteq_i$ and $\preceq_i$, respectively, and let $I(=_i)$ and $I(\equiv_i)$ denote the index of $=_i$ and $\equiv_i$, respectively, i.e., the number of equivalence classes modulo $=_i$ and $\equiv_i$. Then (see Appendix C for the proof):

Lemma 20 *For every $i \leq k$, $I(=_i)$ and $I(\equiv_i)$ can be bounded by an (effectively computable) function in the size of $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$.*

We note that the function constructed in the proof of the lemma grows non-elementary in k. It is open whether this can be improved. Together with the following proposition, Lemma 20 implies that the depth of m_i can be bounded. This shows (*) and concludes the proof of Theorem 12.

Proposition 21 *Let $i < k$, and assume that $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ is a solvable instance of* MESSAGEPROBLEM. *Then, there exists a solution $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ of this instance with $\mathsf{depth}(\overline{m}_i) \leq I(=_i) + \mathsf{depth}(\mathcal{K}) + 1$.*

Proof. Let $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$ be a solution of $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ and assume $\mathsf{depth}(m_i) > I(=_i) + \mathsf{depth}(\mathcal{K}) + 1$. Then there exist unary contexts C, C' and messages x, x' such that C' is not a variable and $m_i = C[C'[x']]$ with $x = C'[x']$, $\mathsf{depth}(x') > \mathsf{depth}(\mathcal{K})$, and $x =_i x'$. Note that $\mathsf{depth}(x') < \mathsf{depth}(x)$. Define $\overline{m}_i := C[x']$.

Since $m_i \in \mathsf{d}(\mathcal{K})$ and x' is a subterm of m_i with $\mathsf{depth}(x') > \mathsf{depth}(\mathcal{K})$, using Lemma 3 it is easy to see that $x' \in \mathsf{d}(\mathcal{K})$ and $\overline{m}_i \in \mathsf{d}(\mathcal{K})$.

Thanks to Lemma 18 and $x =_i x'$, we know that $m_i =_i \overline{m}_i$. Since $q_i^I(m_i) \to^*_{\mathcal{A}_i} m'_i$, there exists a $\overline{m}'_i$ such that $q_i^I(\overline{m}_i) \to^*_{\mathcal{A}_i} \overline{m}'_i$ with $m'_i \preceq_{i+1} \overline{m}'_i$. Using Proposition 19, the problem $(\mathcal{K} \cup \{\overline{m}'_i\}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$ has a solution, say $(\overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$. Thus, $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$. Iterating this argument, the proposition follows. □

7 A Complexity Lower Bound

We prove the following theorem.

Theorem 22 *For TTLA-based protocols,* Attack *is EXPTIME-hard.*

In fact, in the reduction we only use TTLLs. Thus, the theorem also holds for TTLLs.

We first show:

Proposition 23 MessageProblem *is EXPTIME-hard.*

This is done by reduction from the intersection problem for TDTA, which has been shown to be EXPTIME-complete by Seidl [24].

The *finite intersection problem for TDTAs* is defined as follows: Given $k \geq 0$ TDTA $\mathcal{B}_0, \ldots, \mathcal{B}_{k-1}$ with a common alphabet Σ, decide whether there exists a term $t \in \mathcal{T}(\Sigma)$ accepted by $\mathcal{B}_i$ for all $i < k$.

In his proof, Seidl only uses TDTAs over signatures with symbols of rank 0,1, or 2. Using a suitable encoding of trees it is then easy to see that signatures with symbols of rank 0 and one symbol of rank 2 suffice. Let Σ denote such a signature, where we assume that the symbol of rank 2 is the pairing function $<\cdot,\cdot>$. Moreover, let $\mathcal{N}$ be the set of constants in Σ plus a new atomic message a and the atomic message secret.

In what follows, assume $\mathcal{B}_i = (Q_i, \Sigma, I_i, \Delta_i)$ is a TDTA. Given the $\mathcal{B}_i$'s, we define the corresponding message problem as follows. The initial intruder knowledge $\mathcal{K}$ is the set of constants of Σ. The message transducer $\mathcal{A}_i$ will correspond to $\mathcal{B}_i$. The idea is that the intruder guesses a term t that is accepted by all $\mathcal{B}_i$'s ($\mathcal{A}_i$'s). To make sure that the intruder does not send different t's to the different $\mathcal{A}_i$'s, $\mathcal{A}_0$ encrypts t by $\mathsf{enc}_a(t)$. Also, $\mathcal{A}_0$ accepts t only if t is accepted by $\mathcal{B}_0$. This is ensured by simply simulating $\mathcal{B}_0$. Now, the other $\mathcal{A}_i$'s only accept terms encrypted with a and they accept a term of the form $\mathsf{enc}_a(t')$ only if t' is accepted by $\mathcal{B}_i$. The $\mathcal{A}_i$'s simply copy their input to the output. The only term encrypted with a the intruder knows is the one returned by $\mathcal{A}_0$, namely, $\mathsf{enc}_a(t)$. Thus, all the $\mathcal{A}_i$'s must receive the same term $\mathsf{enc}_a(t)$ for the attack to be successful. The last transducer $\mathcal{A}_{k-1}$ returns $<\mathsf{enc}_a(t), \mathsf{secret}>$ if he accepts the term received. Consequently, the intruder obtains the message secret iff he is able to guess a term t accepted by all $\mathcal{B}_i$'s.

Formally, the $\mathcal{A}_i$'s are defined as follows. The states q_i^I, $i < k$, denote new states:

$$\mathcal{A}_0 := (Q_i \cup \{q_0^I\}, \Sigma_{\mathcal{N}}, \{q_0^I\}, \Delta_0')$$

with $\Delta_0' := \Delta_0 \cup \{q_0^I(f(v_0,\ldots,v_{n-1})) \to \mathtt{enc}_a(f(q_0(v_0),\ldots,q_{n-1}(v_{n-1}))) \mid q \in I_0,$ $q(f(v_0,\ldots,v_{n-1})) \to f(q_0(v_0),\ldots,q_{n-1}(v_{n-1})) \in \Delta_i\}$. It is easy to see that $q_0^I(m) \to^*_{\mathcal{A}_0} m'$ iff m is accepted by $\mathcal{B}_0$ and $m' = \mathtt{enc}_a(m)$.

For every $i = 1, \ldots, k-2$:

$$\mathcal{A}_i := (Q_i \cup \{q_i^I\}, \Sigma_{\mathcal{N}}, \{q_i^I\}, \Delta_i \cup \{q_i^I(\mathtt{enc}_a(v)) \to \mathtt{enc}_a(q(v)) \mid q \in I_i\}).$$

Obviously, $q_i^I(m) \to^*_{\mathcal{A}_i} m'$ iff there exists t accepted by $\mathcal{B}_i$ such that $m = m' = \mathtt{enc}_a(t)$.

Finally,

$$\mathcal{A}_{k-1} := (Q_{k-1} \cup \{q_{k-1}^I\}, \Sigma_{\mathcal{N}}, \{q_{k-1}^I\}, \Delta_{k-1}').$$

with $\Delta_{k-1}' := \Delta_{k-1} \cup \{q_{k-1}^I(\mathtt{enc}_a(v)) \to {<}\mathtt{enc}_a(q(v)), \mathsf{secret}{>} \mid q \in I_{k-1}\}$. Analogously to the other transducers, $q_{k-1}^I(m) \to^*_{\mathcal{A}_{k-1}} m'$ iff there exists t accepted by $\mathcal{B}_{k-1}$ such that $m = \mathtt{enc}_a(t)$ and $m' = {<}\mathtt{enc}_a(t), \mathsf{secret}{>}$.

It is now easy to see that $(\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$ has a solution iff there exists a term t accepted by every $\mathcal{B}_i$, $i < k$. This conclude the proof of Proposition 23.

We can basically employ the same argument for Attack. We define the protocol to consist of one principal given by $(\mathcal{A}_0, \ldots, \mathcal{A}_{k-1}, I)$ with $I := \{(q_0^I, \ldots, q_{k-1}^I)\}$. The intruder knowledge is defined as before. Then, this problem is the same as the message problem defined above, and thus, Theorem 22 follows.

An alternative reduction would be to consider a protocol with k principals, where the ith principals performs exactly one receive-send action which is defined by $\mathcal{A}_i$. To avoid that the intruder first sends a message, say the term t, to A_0, and then sends the message $\mathtt{enc}_a(t)$ returned by $\mathcal{A}_0$ immediately to $\mathcal{A}_{k-1}$, one can add a "counter" to the output messages. That is, $\mathcal{A}_0$ outputs $\mathtt{enc}_a({<}a,t{>})$ instead of $\mathtt{enc}_a(t)$, $\mathcal{A}_1$ only accepts a message if it is of this form and outputs $\mathtt{enc}_a({<}a,{<}a,t{>}{>})$, and so forth.

8 Comparison with Models for Non-looping Protocols

In this section, we point out important differences between the tree transducer-based protocol model and models for non-looping protocols.

As mentioned before, the main difference between the two models is that in the former receive-send actions are described by TTLAs whereas, in the latter, they are basically described by single rewrite rules of the form $t \to t'$, applied only once to the input message. Recall from the introduction that, for this reason, we have called the latter kind of receive-send actions non-looping.

The main advantage of the tree transducer-based model is that in this model receive-send actions performing certain iterative processes on the input message can be described, which is needed for iterative protocols (see Appendix D for an example). Such receive-send actions are out of the scope of models for non-looping protocols, as already explained in the introduction. At the same time, the computation power of

TTLAs as regards the "non iterative" transformations is quite close to that of non-looping receive-send actions: As stated in Lemma 2, every transformation described by a rewrite rule $t \to t'$ can also be carried out by a TTLA as long as t is a linear term. The word transducers considered in [15] cannot even describe all rewrite rules where *both* t and t' are linear, and thus, they are much more restricted than TTLAs.

One restriction of the tree transducer-based model compared to models for non-looping protocols is that receive-send actions in which messages of arbitrary size are compared for equality cannot be modeled. This is, however, possible with non-looping receive-send actions of the form $t \to t'$ if t is non-linear. Theorem 8 implies that extending TTLAs to include this feature makes ATTACK undecidable. Another restriction is that, roughly speaking, TTLA-based principals have only finite memory whereas principals described by non-looping receive-send actions have unbounded memory: In TTLA-based principals the relation I (see Definition 5) allows to convey a finite amount of information from one receive-send action to another while in models for non-looping protocols the different rewrite rules $t \to t'$, each representing one receive-send action, may share variables, which can be substituted by messages of arbitrary size, and thus, an unbounded amount of information can be transmitted between different receive-send actions. Theorem 9 indicates that extending TTLAs with unbounded memory leads to undecidability of ATTACK.

As a remedy for the restrictions of the tree transducer-based model one can try to find weaker forms of equality tests and other ways to incorporate unbounded memory (see, e.g., [7]). Also, assuming messages to be typed, which is often the case in real protocols, can help to avoid general equality tests and might make unbounded memory superfluous [14].

9 Conclusion

We have proposed a tree-transducer based protocol model designed to describe iterative protocols and based on this model we have drawn a line between decidability and undecidability of security for such protocols.

It remains to establish a tight complexity bound for security in the TTLA-based protocol model. When simplifying the MESSAGEPROBLEM (Section 6), by basically ignoring the intruder, the problem of deciding whether a certain output is produced by a composition of TTLAs can be seen as an instance of the simplified MESSAGEPROBLEM — the complexity of composing tree transducers has been investigated in [2, 16]. It might be insightful to further study the connections between the two problems. Also, a promising future direction is to combine the tree transducer-based model with the models for non-looping protocols. Finally, different extensions also considered for non-looping protocols such as complex keys shall be investigated.

References

[1] R.M. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. Technical Report RR-4147, INRIA, 2001.

[2] B. S. Baker. Generalized syntax directed translation, tree transducers, and linear space. *SIAM Journal on Computing*, 7(3):376–391, 1978.

[3] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Automata, Languages and Programming, 28th International Colloquium (ICALP 2001)*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681. Springer-Verlag, 2001.

[4] J. Bryans and S.A. Schneider. CSP, PVS, and a Recursive Authentication Protocol. In *DIMACS Workshop on Formal Verification of Security Protocols*, 1997.

[5] 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.

[6] J. Clark and J. Jacob. *A Survey of Authentication Protocol Literature*, 1997. Web Draft Version 1.0 available from http://citeseer.nj.nec.com/.

[7] H. Comon, V. Cortier, and J. Mitchell. Tree automata with one memory, set constraints, and ping-pong protocols. In *Automata, Languages and Programming, 28th International Colloquium (ICALP 2001)*, volume 2076 of *Lecture Notes in Computer Science*, pages 682–693, 2001.

[8] H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technolog*, 2002. To appear.

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

[10] J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303, 1976/1977.

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

[12] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics — Formal Models Based on Tree Transducers.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 1998.

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

[14] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW'00)*, pages 255–268. IEEE Computer Society, 2000.

[15] 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.

[16] S. Maneth. The Complexity of Compositions of Deterministic Tree Transducers. In M. Agrawal and A. Seth, editors, *FSTTCS 2002: Foundations of Software Technology and Theoretical Computer Science — 22nd Conference*, volume 2556 of *Lecture Notes in Computer Science*, pages 265–276. Springer-Verlag, 2002.

[17] C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P. Degano, editor, *Proceedings of the First Workshop on Issues in the Theory of Security (WITS'00)*, pages 87–92, 2000.

[18] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of 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 *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM Press, 2001.

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

[21] L.C. Paulson. Mechanized Proofs for a Recursive Authentication Protocol. In *10th IEEE Computer Security Foundations Workshop (CSFW-10)*, pages 84–95, 1997.

[22] O. Pereira and J.-J. Quisquater. A Security Analysis of the Cliques Protocols Suites. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 73–81, 2001.

[23] M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop (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.

A Proof of Lemma 18

First, we note that $\preceq_{\mathcal{G}}$ is closed under substitution:

Remark 24 *For messages $t_0, \ldots, t_{n-1}$, $t'_0, \ldots, t'_{n-1}$ and an n-ary context C: $t_i \preceq_{\mathcal{G}} t'_i$ for every $i < n$ implies $C[t_0, \ldots, t_{n-1}] \preceq_{\mathcal{G}} C[t'_0, \ldots, t'_{n-1}]$.*

Now, Lemma 18 is proved by induction on $i \leq k$, simultaneously for $\preceq_i$ and $\sqsubseteq_i$.

Base case

The base case, $i = k$, amounts to showing that $\preceq$ is closed under substitution:

Lemma 25 *Let $x_0, \ldots, x_{n-1}, x'_0, \ldots, x'_{n-1} \in \mathcal{M}$ be messages and C be an n-ary context. If $x_j \preceq x'_j$ for every $j < n$, then $C[x_0, \ldots, x_{n-1}] \preceq C[x'_0, \ldots, x'_{n-1}]$.*

PROOF. Obviously, the statement holds if C is an atomic message or a variable. We now show that the statement holds for every $C = f(v_0, \ldots, v_{n-1})$ with $f \in \Sigma_{\mathcal{N}}$ and distinct variables $v_0, \ldots, v_{n-1}$. By structural induction, the lemma then easily follows. In what follows, let $N \subseteq \mathcal{N}$.

Assume $C = <v_0, v_1>$. We show $\mathsf{an}_{\mathcal{N}}(<x_0, x_1>, N) \subseteq \mathsf{an}_{\mathcal{N}}(<x'_0, x'_1>, N)$. Let φ be the the following mapping φ on $\mathcal{N}$:

$$\varphi(N') := \mathsf{an}_{\mathcal{N}}(x_0, \mathsf{an}_{\mathcal{N}}(x_1, N'))$$

for every $N' \subseteq \mathcal{N}$. Analogously, we define ψ where x_0 and x_1 are replaced by x'_0 and x'_1, respectively. Note that φ and ψ are monotonic since $\mathsf{an}_{\mathcal{N}}(x, \cdot)$ is a monotonic operator for every message x.

It is easy to see that $\mathsf{an}_{\mathcal{N}}(<x_0, x_1>, N)$ is the least fixed point of φ containing N, i.e., the set $\bigcup_{l \geq 0} \varphi^l(N)$ with $\varphi^0(N) := N$ and $\varphi^{l+1}(N) := \varphi(\varphi^l(N))$. Analogously, $\mathsf{an}_{\mathcal{N}}(<x'_0, x'_1>, N)$ is the least fixed point of ψ containing N.

Now, $x_j \preceq x'_j$ implies $\mathsf{an}_{\mathcal{N}}(x_j, N') \subseteq \mathsf{an}_{\mathcal{N}}(x'_j, N')$ for $j < 2$ and $N' \subseteq \mathcal{N}$. Thus, $\varphi(N') \subseteq \psi(N')$ for every $N' \subseteq \mathcal{N}$. Hence,

$$\mathsf{an}_{\mathcal{N}}(<x_0, x_1>, N) = \bigcup_{l \geq 0} \varphi^l(N) \subseteq \bigcup_{l \geq 0} \psi^l(N) = \mathsf{an}_{\mathcal{N}}(<x'_0, x'_1>, N).$$

Now, assume that $C = \mathtt{enc}_a(v)$. If $a \notin N$, it follows $\mathsf{an}_{\mathcal{N}}(\mathtt{enc}_a(x_0), N) = N = \mathsf{an}_{\mathcal{N}}(\mathtt{enc}_a(x'_0))$. Assume $a \in N$. Then,

$$\mathsf{an}_{\mathcal{N}}(\mathtt{enc}_a(x_0), N) = \mathsf{an}_{\mathcal{N}}(x_0, N) \subseteq \mathsf{an}_{\mathcal{N}}(x'_0, N) = \mathsf{an}_{\mathcal{N}}(\mathtt{enc}_a(x'_0), N).$$

For $C = \mathtt{hash}(v)$ we obtain $\mathsf{an}_{\mathcal{N}}(\mathtt{hash}(x_0), N) = N = \mathsf{an}_{\mathcal{N}}(\mathtt{hash}(x'_0), N)$. □

From this it immediately follows that $\preceq_k$ is closed under substitution. Note that for $\sqsubseteq_i$ nothing is to show since for $i = k$, $\sqsubseteq_i$ is not defined.

Induction step

We first need some notation: Let $\mathsf{an}_c(T)$ denote the closure of T under decomposition, i.e., $\mathsf{an}_c(T)$ denotes the smallest set S such that $T \subseteq S$ and if $<m, m'> \in S$, then $m, m' \in S$. Given a term t and $N \subseteq \mathcal{N}$, we say that a subterm t' of t is N-*accessible*[1] if

1. $t' \in \mathsf{an}_c(t)$, or
2. there exists $\mathtt{enc}_a(t'') \in \mathsf{an}_c(t)$ for some $a \in N$ such that t' is N-accessible in t''.

[1]This notion was also used in [1]

Define $m := C[x_0, \ldots, x_{n-1}]$ and $m' := C[x'_0, \ldots, x'_{n-1}]$. Assume $i < k$.

Claim I. From $x_j \sqsubseteq_i x'_j$ for every $j < n$ it follows $m \sqsubseteq_i m'$.

Proof of Claim I. By Remark 24, it follows $m \preceq_{\mathcal{G}} m'$.

The remainder of the proof is by structural induction on C. If C is an atomic message or a variable, the statement follows immediately.

Assume that C is neither a variable nor an atomic message, i.e., $C = f(C_0, \ldots, C_{r-1})$ for some $f \in \Sigma_{\mathcal{N}}$ and contexts C_i. Let $q \in Q_i$ and $y \in \mathcal{M}$ such that $q(m) \to^*_{\mathcal{A}_i} y$. We show that there exists y' with $q(m') \to^*_{\mathcal{A}_i} y'$ and $y \preceq_{i+1} y'$.

Assume that in the computation $q(m) \to^*_{\mathcal{A}_i} y$ the first transition applied is

$$\tau : q(f(v_0, \ldots, v_{r-1})) \xrightarrow{p_0,\ldots,p_{r-1}} C'[q_0(v_{i_0}), \ldots, q_{r'-1}(v_{i_{r'-1}})].$$

Then, the computation is of the form:

$$q(m) \to_{\mathcal{A}_i} C'[q_0(m_{i_0}), \ldots, q_{r'-1}(m_{i_{r'-1}})] \to^*_{\mathcal{A}_i} y = C'[y_0, \ldots, y_{r'-1}]$$

with $m_j = C_j[x_0, \ldots, x_{n-1}] \in \mathcal{L}(\mathcal{G}_{p_j})$, $j < r$, and $q_j(m_{i_j}) \to^*_{\mathcal{A}_i} y_j$, $j < r'$. By definition, $m' = f(m'_0, \ldots, m'_{r-1})$ with $m'_j = C_j[x'_0, \ldots, x'_{n-1}]$. By Remark 24, $m'_j \in \mathcal{L}(\mathcal{G}_{p_j})$. Thus, τ can be applied to $q(m')$ yielding

$$q(m') \to_{\mathcal{A}_i} C'[q_0(m'_{i_0}), \ldots, q_{r'-1}(m'_{i_{r'-1}})].$$

By structural induction, $m_j \sqsubseteq_i m'_j$, $j < r$, and thus, it follows that there exist y'_j, $j < r'$, with $q_j(m'_{i_j}) \to^*_{\mathcal{A}_i} y'_j$ and $y_j \preceq_{i+1} y'_j$. Then, the induction on i yields $y \preceq_{i+1} C'[y'_0, \ldots, y'_{r'-1}] =: y'$ and we know $q(m') \to^*_{\mathcal{A}_i} y'$. This completes the proof of Claim I.

Claim II. From $x_j \preceq_i x'_j$ for every $j < n$ it follows $m \preceq_i m'$.

Proof of Claim II. By Lemma 25, we have $m \preceq m'$. Since $x_j \preceq_i x'_j$ implies $x_j \preceq_{i+1} x'_j$, induction on i yields $m \preceq_{i+1} m'$. Moreover, $x_j \preceq_i x'_j$ implies $x_j \sqsubseteq_i x'_j$, and by Claim I, $m \sqsubseteq_i m'$.

Let $N \subseteq \mathcal{N}$ and $x \in \mathsf{an}(m, N)$. We need to show that there exists $x' \in \mathsf{an}(m', N)$ with $x \sqsubseteq_i x'$. We distinguish two cases:

- There exists a subterm C' of C such that $x = C'[x_0, \ldots, x_{n-1}]$. Define $x' := C'[x'_0, \ldots, x'_{n-1}]$. We know that C' is $\mathsf{an}_{\mathcal{N}}(m, N)$-accessible in C. Using $m \preceq m'$, and thus, $\mathsf{an}_{\mathcal{N}}(m, N) \subseteq \mathsf{an}_{\mathcal{N}}(m', N)$, it follows that C' is $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in C. In particular, $x' \in \mathsf{an}(m', N)$. Since $x_j \sqsubseteq_i x'_j$, Claim I implies $x \sqsubseteq_i x'$.
- There exists $j < n$ with x subterm of x_j, and x and x_j $\mathsf{an}_{\mathcal{N}}(m, N)$-accessible in m. Now, $m \preceq m'$ implies that x'_j is $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in m'. Also, $x \in \mathsf{an}(x_j, \mathsf{an}_{\mathcal{N}}(m, N))$. Because $x_j \preceq_i x'_j$, there exists $x' \in \mathsf{an}(x'_j, \mathsf{an}_{\mathcal{N}}(m, N))$ with $x \sqsubseteq_i x'$. Finally, $m \preceq m'$ implies $x \in \mathsf{an}(x'_j, \mathsf{an}_{\mathcal{N}}(m', N))$, and given that x'_j is $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in m' it follows $x' \in \mathsf{an}(m', N)$.

B Proof of Proposition 19

The proof of Proposition 19 is by induction on $i \le k$.

Base case

The base case, $i = k$, is a consequence of the following lemma.

Lemma 26 *For all $m, m' \in \mathcal{M}$, $\mathcal{K} \subseteq \mathcal{M}$, if $m \preceq m'$, then $\mathsf{an}_{\mathcal{N}}(m, \mathcal{K}) \subseteq \mathsf{an}_{\mathcal{N}}(m', \mathcal{K})$.*

PROOF. The proof is similar to the case $C = \langle v_0, v_1 \rangle$ in the proof of Lemma 25. We define the following mapping φ on atomic messages:

$$\varphi(N) := \mathsf{an}_{\mathcal{N}}(m, \mathsf{an}_{\mathcal{N}}(\mathcal{K}, N)).$$

The mapping ψ is defined analogously where m is replaced by m'. Note that φ and ψ are monotonic since $\mathsf{an}_{\mathcal{N}}(x, \cdot)$ is monotonic. It is easy to see that $\mathsf{an}_{\mathcal{N}}(m, \mathcal{K})$ is the least fixed point of φ, i.e., the set $\bigcup_{l \ge 0} \varphi^l(\emptyset)$. The same holds for ψ if m is replaced by m'. Since $m \preceq m'$, $\varphi(N) \subseteq \psi(N)$ for every $N \subseteq \mathcal{N}$. From this the claim follows easily. $\square$

Induction step

Assume $i < k$. Define $N := \mathsf{an}_{\mathcal{N}}(\mathcal{K})$. Let $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$ be a solution of $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$. Thus, $m_i \in \mathsf{d}(\mathcal{K})$. Since $\mathsf{d}(\mathcal{K}) = \mathsf{syn}(\mathsf{an}(\mathcal{K}))$, the derivation of m_i can be carried out by an analysis and a synthesis phase, in this order. Let $\{x_0, \ldots, x_{n-1}\}$ be the multiset of messages in $\mathsf{an}(m, N)$ used in the synthesis phase of m_i's derivation such that there exists an n-ary context C with $m_i = C[x_0, \ldots, x_{n-1}]$. Since $m \preceq_i \overline{m}$, there exist messages $\overline{x}_j \in \mathsf{an}(\overline{m}, N)$ such that $x_j \sqsubseteq_i \overline{x}_j$ for every $j < n$. Define $\overline{m}_i := C[\overline{x}_0, \ldots, \overline{x}_{n-1}]$. To show $\overline{m}_i \in \mathsf{d}(\overline{\mathcal{K}})$ replace in the synthesis of m_i the message x_j by $\overline{x}_j$. This yields $\overline{m}_i$. This construction of $\overline{m}_i$ can in fact be carried out in $\overline{\mathcal{K}}$ since $N \subseteq \mathsf{an}_{\mathcal{N}}(\overline{m}, \mathcal{K})$, every subterm of some term in $\mathcal{K} \setminus \{m\}$ used in the synthesis belongs to $\mathsf{an}(\mathcal{K} \setminus \{m\}, N)$, and every subterm of $\overline{m}$ used in the synthesis is some $\overline{x}_j \in \mathsf{an}(\overline{m}, N)$. Thus, $\overline{m}_i \in \mathsf{d}(\overline{\mathcal{K}})$.

Lemma 18 implies $m_i \sqsubseteq_i \overline{m}_i$. Hence, with $q_i^I(m_i) \to^*_{\mathcal{A}_i} m'_i$ there exists $\overline{m}'_i$ such that $q_i^I(\overline{m}_i) \to^*_{\mathcal{A}_i} \overline{m}'_i$ and $m'_i \preceq_{i+1} \overline{m}'_i$.

By induction on i, the solvability of $(\mathcal{K} \cup \{m'_i\}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$ implies the one for $(\mathcal{K} \cup \{\overline{m}'_i\}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$. Moreover, since $m \preceq_i \overline{m}$ implies $m \preceq_{i+1} \overline{m}$, induction yields that $(\overline{\mathcal{K}} \cup \{\overline{m}'_i\}, \mathcal{A}_{i+1}, \ldots, A_{k-1})$ has a solution, say $(\overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$. Finally, we can conclude that $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ solves $(\overline{\mathcal{K}}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$.

C Proof of Lemma 20

If $\le$ is a quasi-ordering on a set S, then the relation $\equiv$ with $a \equiv b$ iff $a \le b$ and $b \le a$ for all $a, b \in S$ is an equivalence relation on S. The *equivalence class* of a

w.r.t. $\equiv$ is denoted $[a]_\equiv := \{b \mid a \equiv b\}$. The *index* $I(\equiv)$ of $\equiv$ is the number of different equivalence classes over S w.r.t. $\equiv$.

In what follows, let $\equiv$, $\equiv_\mathcal{G}$, $=_i$, and $\equiv_i$, denote the equivalence relations corresponding to $\preceq$, $\preceq_\mathcal{G}$, $\sqsubseteq_i$, and $\preceq_i$.

We show that the index of these equivalence relations is finite. For $\equiv$ and $\equiv_\mathcal{G}$ the proof is straightforward.

Lemma 27

$$\begin{array}{rcl} I(\equiv) & \leq & 2^{2^{|\mathcal{N}|}\cdot|\mathcal{N}|}, \\ I(\equiv_\mathcal{G}) & \leq & 2^{|Q|}, \end{array}$$

where Q denotes the set of states of $\mathcal{G}$.

The following lemma generalizes this to $=_i$ and $\equiv_i$. Note that $=_i$ is only defined for $i < k$.

Lemma 28 *For every $i < k$:*

$$\begin{array}{rcl} I(\equiv_k) & \leq & I(\equiv) \\ I(=_i) & \leq & I(\equiv_\mathcal{G}) \cdot 2^{I(\equiv_{i+1})\cdot 2^{|Q_i|}}, \\ I(\equiv_i) & \leq & I(\equiv_{i+1}) \cdot I(=_i) \cdot 2^{I(=_i)\cdot 2^{|\mathcal{N}|}}. \end{array}$$

Proof. The statement is proved for both equivalence relations simultaneously by induction on i. The case $i = k$, follows from Lemma 27. Note that for the computation preserving equivalence relation nothing is to show. Now, assume that $i < k$.

Claim I. The index of $=_i$ is bounded as stated in the lemma.

Proof of Claim I. We introduce a new equivalence relation on tuples (x, y) with $x, y \in \mathcal{M}$. For every $x, x', y, y' \in \mathcal{M}$ define: $(x, y) =_i^t (x', y')$ iff

- $y \equiv_{i+1} y'$, and
- $q(x) \rightarrow^*_{\mathcal{A}_i} y$ iff $q(x') \rightarrow^*_{\mathcal{A}_i} y'$ for every $q \in Q_i$.

To prove that $=_i^t$ has finite index, consider the mapping φ_i^t which takes every tuple (x, y) to the tuple $([y]_{\equiv_{i+1}}, \{q \mid q(x) \rightarrow^*_{\mathcal{A}_i} y\})$. It is easy to see that $\varphi_i^t(x, y) = \varphi_i^t(x', y')$ implies $(x, y) =_i^t (x', y')$. From this, it immediately follows:

$$I(=_i^t) \leq I(\equiv_{i+1}) \cdot 2^{|Q_i|}.$$

To show the bound for $=_i$, define $M_{i,x} := \{[(x, y)]_{=_i^t} \mid y \in \mathcal{M}\}$. We show that for every $x, x' \in \mathcal{M}$, $([x]_{\equiv_\mathcal{G}}, M_{i,x}) = ([x']_{\equiv_\mathcal{G}}, M_{i,x'})$ implies $x =_i x'$: Obviously, $x \equiv_\mathcal{G} x'$. Now, let $q \in Q_i$ and $y \in \mathcal{M}$ with $q(x) \rightarrow^*_{\mathcal{A}_i} y$. We know $[(x, y)]_{=_i^t} \in M_{i,x} = M_{i,x'}$. Thus, there exists $y' \in \mathcal{M}$ such that $[(x, y)]_{=_i^t} = [(x', y')]_{=_i^t}$. Consequently, by definition of $=_i^t$, $y \equiv_{i+1} y'$ and $q(x') \rightarrow^*_{\mathcal{A}_i} y'$. This shows $x \sqsubseteq_i x'$. By symmetry, $x' \sqsubseteq_i x$.

From this it immediately follows that

$$I(=_i) \leq I(\equiv_{\mathcal{G}}) \cdot 2^{I(=_i^t)} \leq I(\equiv_{\mathcal{G}}) \cdot 2^{I(\equiv_{i+1}) \cdot 2^{|Q_i|}}.$$

This completes the proof of Claim I.

Claim II. The index of $\equiv_i$ is bounded as stated in the lemma.

Proof of Claim II. For $m \in \mathcal{M}$ and $N \subseteq \mathcal{N}$ let $M_{i,m,N} := \{[x]_{=_i} \mid x \in \mathsf{an}(m, N)\}$. Define φ_i to be the mapping that takes every message $m \in \mathcal{M}$ to $([m]_{\equiv_{i+1}}, [m]_{=_i}, (M_{i,m,N} \mid N \subseteq \mathcal{N}))$. The size of the range of φ_i is bounded by $I(\equiv_{i+1}) \cdot I(=_i) \cdot 2^{I(=_i) \cdot 2^{|\mathcal{N}|}}$. Thus, to prove the claim it remains to show that $\varphi_i(m) = \varphi_i(m')$ implies $m \equiv_i m'$, for every $m, m' \in \mathcal{M}$:

Assume $\varphi_i(m) = \varphi_i(m')$. Clearly, this implies $m \sqsubseteq_i m'$ and $m \preceq_{i+1} m'$, and thus, $m \preceq m'$. Now, let $N \subseteq \mathcal{N}$, $x \in \mathsf{an}(m, N)$. Thus, $[x]_{=_i} \in M_{i,m,N} = M_{i,m',N}$. Consequently, there exists $x' \in \mathsf{an}(m', N)$ with $[x']_{=_i} = [x]_{=_i}$. In particular, $x \sqsubseteq_i x'$. This shows $m \preceq_i m'$. By symmetry, $m' \preceq_i m$. □

From this, Lemma 20 follows immediately.

D Modeling the Recursive Authentication Protocol

In Appendix D.1, we first provide an informal description of the recursive authentication protocol (RA protocol). Since this protocol uses message authentication codes (MACs), in Appendix D.2 it is shown how MACs can be modeled. Then, Appendix D.3 contains the tree transducer-based model of the RA protocol.

D.1 Informal Description of the RA Protocol

The RA protocol was proposed by Bull and Otway [5] and it extends the authentication protocol by Otway and Rees [20] in that it allows to establish session keys between an a priori unbounded number of principals in one protocol run. Our description of the RA protocol follows Paulson [21].

In the RA protocol one assumes that a key distribution server S shares long-term keys with the principals. In Figure 1 a typical protocol run is depicted. In this run, A wants to establish a session key with B and B wants to establish a session key with C. The number of principals involved in a protocol run is not bounded. In particular, C could send a message to some principal D in order to establish a session key with D and D could continue and send a message to E and so on. In the protocol run depicted in Figure 1, we assume that C does not want to talk to another principal and therefore sends a message to the key distribution server S, who is involved in every protocol run.

In Figure 1, K_a denotes the long-term keys shared between A and S. Similarly, K_b and K_c are the long-term keys shared between B, C, and S, respectively. With N_a, N_b, and N_c we denote nonces (i.e., random numbers) generated by A, B, and C, respectively. Finally, K_{ab}, K_{bc}, and K_{cs} are the session keys generated by the server and used by the principals for secure communication between A and B, B and C, and C and S, respectively. The numbers (1. – 6.) attached to the messages only indicate the order in which the messages are sent and do not belong the protocol. The message $\mathtt{hash}_a(m)$ stands for the message authentication code (MAC) of m w.r.t. the key a concatenated with m. For instance, $\mathtt{hash}_a(m)$ could be the message $<\mathtt{hash}(<a, m>), m>$. However, there are other ways to represent the MAC of m w.r.t. a than by $\mathtt{hash}(<a, m>)$. We abbreviate messages of the form $<m_0, \ldots, <m_{n-1}, m_n> \cdots>$ by $m_0 \cdots m_n$.

We now take a closer look at the messages exchanged between the principals in the order they are sent: In the first messages (1.), principal A indicates that she requests a session key from the server for secure communication with B. The symbol "$-$" says that this message started the protocol run. Now, in the second message (2.), B sends something similar to C but with A's message instead of "$-$", indicating that he wants to share a session key with C. As mentioned, this step could be repeated as many times as desired, yielding an ever-growing stack of requests. The process is terminated if one principal contacts S. In this example, we assume that C does not request another session key, and therefore, sends the message received from B to S (3.). This message is now processed by S.

The outer hash (request) indicates that C has called S. Therefore, the server generates a fresh key K_{cs} intended to be used as a session key between C and S.[2] Then, S prepares a certificate $\mathtt{enc}_{K_c}(K_{cs}SN_c)$ which together with the certificates prepared later will be sent to C. Finally, S remembers C's name and the nonce N_c. Now, the request is discarded and the next one is examined in the same way. This request indicates that B has called C, i.e., the principal's name S has stored from the previous request. As before, S generates a fresh key K_{bc} intended to be used as a session key between B and C, and prepares two certificates $\mathtt{enc}_{K_c}(K_{bc}BN_c)$ and $\mathtt{enc}_{K_b}(K_{bc}CN_b)$ which are added to the certificate prepared before. Note that to generate the former certificate, S uses N_c, which was previously stored by S. Now, S stores B's name and the nonce N_b. The last request $\mathtt{hash}_{K_a}(ABN_a-)$ is processed in the same way. The symbol "$-$" indicates that no more requests follow. Having prepared all certificates, the server sends all of them to C (4.). The line break is only for layout purposes and does not have any meaning in the protocol.

Principal C accepts the first two certificates, extracts the two session keys, and forwards the rest of the message to his predecessor in the chain (5.). Then, B does the same, and forwards the last certificate to A (6.)

D.2 Modeling Message Authentication Codes

One can extend the TTLA-based protocol model to include different cryptographic primitives. In this section, we consider the message authentication code (MAC), which in Appendix D.3 is used to model the recursive authentication protocol. The purpose of a MAC is to guarantee integrity of messages.

We will write the MAC over a message m with key a as $\mathtt{hash}_a(m)$. This message does not only represent the digest of m but also the message m itself. So on seeing

[2]Note that K_{cs} is redundant since C and S already share a key. But including it allows to treat the last principal in the chain like all others, except the first, who only receives one session key.

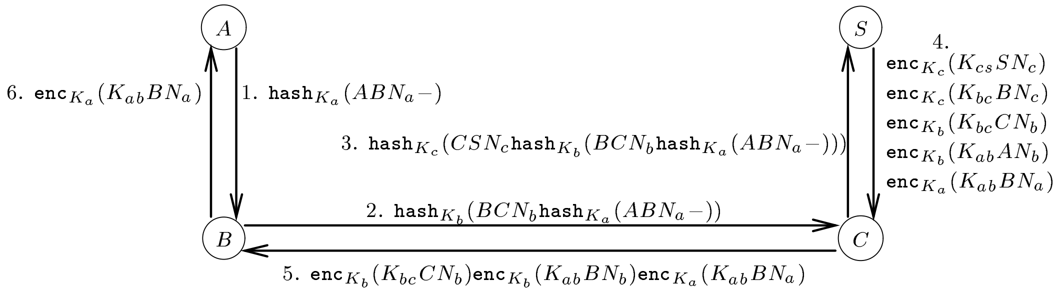

Figure 1: A Run of the Recursive Authentication Protocol

$\mathtt{hash}_a(m)$, everyone can read m but only those principals who know the key a can create $\mathtt{hash}_a(m)$. In particular, the intruder cannot change m, say to m', without knowledge of a because to create $\mathtt{hash}_a(m')$ the intruder would need to know a.

Formally, we extend the TTLA-based protocol model as follows: The signature $\Sigma_{\mathcal{N}}$ will include for every $a \in \mathcal{N}$ the unary symbol $\mathtt{hash}_a$. To the definition of $\mathsf{d}(\mathcal{K})$ (see Section 4), and thus, the definition of the intruder, the following conditions are added:

- If $\mathtt{hash}_a(m) \in \mathsf{d}(\mathcal{K})$, then $m \in \mathsf{d}(\mathcal{K})$.
- If $m \in \mathsf{d}(\mathcal{K})$ and $a \in \mathsf{d}(\mathcal{K}) \cap \mathcal{N}$, then $\mathtt{hash}_a(m) \in \mathsf{d}(\mathcal{K})$.

The first condition allows the intruder to look "inside" a MAC and the second allows him to create MACs himself given that he knows the corresponding key. All undecidability, decidability, and complexity theoretic results carry over to this extended intruder model in a straightforward way. Basically, only the proof of Lemma 25 needs little change.

Instead of introducing a new kind of function symbol and extending the intruder, one could model MACs by messages of the form $<\mathtt{hash}(<a,m>),m>$, which does not require to change the intruder model. However, on receiving this message, a principal would need to check whether the message m inside of the hash and the one outside coincide. But this requires to check equality of messages of arbitrary size, which is not possible with TTLA-based protocols and as stated in Theorem 8, adding this capability leads to undecidability of Attack. This is the reason why this alternative of modeling MACs is not chosen here.

D.3 The TTLA-based Protocol Model

We now provide a formal description of the the RA protocol in the TTLA-based protocol model extended with MACs as discussed in Appendix D.2. For convenience, the TTLAs are described as TTLLs. By Lemma 1, a TTLL can easily be turned into a TTLA.

First we note that although in the RA protocol the number of receive-send actions performed in one protocol run is unbounded, in our model we assume a fixed bound — extending the TTLA-based protocol model to handle an unbounded number of receive-send actions would lead to undecidability, just as for an unbounded number of sessions, as mentioned in Section 5. Nevertheless, even with such a fixed bound it is still necessary to model iterative processes. In the RA protocol, for instance, the intruder can generate an unbounded sequence of requests which must be processed by the server. We note that in other protocols in which the number of receive-send action in a protocol run is fixed, iterative processes may also occur independent of the intruder. One example is IKE [13].

In what follows, 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 nonce sent by P_i in the request message is denoted N_i, $i < n$.

D.3.1 Modeling the Agents

An agent P_i, $i < n$, performs two receive-send actions — one request action and one receive action — and is given by the tuple $(\mathcal{A}_0^i, \mathcal{A}_1^i, I^i)$. The different component are defined next.

The message transducer $\mathcal{A}_0^i$ for sending the request message consists of the following transitions:

$$\begin{array}{rcl} (\mathsf{request}, \perp, P_{j'})(init) & \to & \mathtt{hash}_{K_i}(P_i P_{j'} N_i -), \\ (\mathsf{request}, P_j, P_{j'})(\mathtt{hash}_{a_0}(P_j P_i a_1 v) & \to & \mathtt{hash}_{K_i}(P_i P_{j'} N_i \mathtt{hash}_{a_0}(P_j P_i a_1 \mathsf{copy}(v)), \text{ and} \\ \mathsf{copy}(f(v_0, \ldots, v_{l-1})) & \to & f(\mathsf{copy}(v_0), \ldots, \mathsf{copy}(v_{l-1})) \end{array}$$

where $j' \leq n$, $j < n$, $a_0, a_1 \in \mathcal{N}$, f is an l-ary symbol in $\Sigma_{\mathcal{N}}$ for some l, and $init \in \Sigma_{\mathcal{N}}$ is some constant known to the intruder. The first transition is applied if the P_i initiates a protocol run and calls $P_{j'}$. The second transition is applied if P_i is called by P_j and sends a message to $P_{j'}$. The last transition only serves to copy the input message to the output. The initial states of $\mathcal{A}_0^i$ are $(\mathsf{request}, \perp, P_{j'})$ and $(\mathsf{request}, P_j, P_{j'})$ for every $j' \leq n$ and $j < n$.

The transitions of $\mathcal{A}_1^i$, the message transducer for receiving the session keys, are:

$$\begin{array}{rcl} (\mathsf{key}, \perp, P_{j'})(\mathtt{enc}_{K_i}(a P_{j'} N_i)) & \to & \mathtt{enc}_a(\mathsf{secret}) \text{ and} \\ (\mathsf{key}, P_j, P_{j'})(\mathtt{enc}_{K_i}(a_0 P_{j'} N_i)\mathtt{enc}_{K_i}(a_1 P_j N_i)) & \to & \mathtt{enc}_{a_0}(\mathsf{secret})\mathtt{enc}_{a_1}(\mathsf{secret}), \end{array}$$

where $j' \leq n$, $j < n$, and $a_0, a_1 \in \mathcal{N}$. The first transition is applied if P_i initiated the protocol run for communication with $P_{j'}$. The session key received is a and secret is encrypted with a to check whether the intruder can get hold of a. In the second transition, P_i called $P_{j'}$ and was called by P_j. The session keys for communication with these principals are a_0 and a_1, respectively. Again, secret is encrypted with these keys. All the states occurring in $\mathcal{A}_1^i$ are initial states.

It remains to define I^i. We want to guarantee that P_i remembers who he called and who wants to communicate with P_i. Therefore, we set

$$\begin{array}{rcl} I^i & := & \{((\mathsf{request}, \perp, P_{j'}), (\mathsf{key}, \perp, P_{j'})) \mid j' \leq n\} \cup \\ & & \{((\mathsf{request}, P_j, P_{j'}), (\mathsf{key}, P_j, P_{j'})) \mid j' \leq n, j < n\}. \end{array}$$

D.3.2 Modeling the Server

Since the server $S = P_n$ performs only one receive-send action, it can be described by a single message transducer, which we call $\mathcal{A}_n$.

The states of $\mathcal{A}_n$ consist of three components. The first takes the values start and read. In state start, $\mathcal{A}_n$ reads the outer request of the message received and checks whether this request is really addressed to S, and if so, generates the first certificate. In state read, $\mathcal{A}_n$ processes the rest of the requests. In the second component, $\mathcal{A}_n$ memorizes whose certificates are to be generated, and the third component stores the corresponding nonce.

The transitions of $\mathcal{A}_n$ are specified as follows:

$$\begin{array}{rcl}
(\mathsf{start}, \bot, \bot)(\mathtt{hash}_{K_i}(P_i P_n a-)) & \to & \mathtt{enc}_{K_i}(K_{in} P_n a), \\
(\mathsf{start}, \bot, \bot)(\mathtt{hash}_{K_i}(P_i P_n a v)) & \to & <\mathtt{enc}_{K_i}(K_{in} P_n a), (\mathsf{read}, P_i, a)(v)>, \\
(\mathsf{read}, P_i, a_0)(\mathtt{hash}_{K_{i'}}(P_{i'} P_i a_1 -)) & \to & <\mathtt{enc}_{K_i}(K_{i'i} P_{i'} a_0), \mathtt{enc}_{K_{i'}}(K_{i'i} P_i a_1)>, \text{ and}
\end{array}$$

$$\begin{array}{c}
(\mathsf{read}, P_i, a_0)(\mathtt{hash}_{K_{i'}}(P_{i'} P_i a_1 v)) \\
\downarrow \\
<\mathtt{enc}_{K_i}(K_{i'i} P_{i'} a_0), <\mathtt{enc}_{K_{i'}}(K_{i'i} P_i a_1), (\mathsf{read}, P_{i'}, a_1)(v)>>,
\end{array}$$

where $i, i' < n$ and $a, a_0, a_1 \in \mathcal{N}$. The state $(\mathsf{start}, \bot, \bot)$ is the only initial state of the server.