On the Decidability of Cryptographic Protocols with Open-ended Data Structures

Ralf Küsters

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel, Germany
kuesters@ti.informatik.uni-kiel.de

Abstract. Formal analysis of cryptographic protocols has mainly concentrated on protocols with closed-ended data structures, where closed-ended data structure means that the messages exchanged between principals have fixed and finite format. However, in many protocols the data structures used are open-ended, i.e., messages have an unbounded number of data fields. Formal analysis of protocols with open-ended data structures is one of the challenges pointed out by Meadows. This work studies decidability issues for such protocols. We propose a protocol model in which principals are described by transducers, i.e., finite automata with output, and show that in this model security is decidable and PSPACE-hard in presence of the standard Dolev-Yao intruder.

1 Introduction

Formal methods are very successful in analyzing the security of cryptographic protocols. Using these methods, many flaws have been found in published protocols. By now, a large variety of different methods and tools for cryptographic protocol analysis is available (see [16] for an overview). In particular, for different interesting classes of protocols and intruders, security has been shown to be decidable, usually based on the Dolev-Yao model [7] (see the paragraph on related work).

Previous work has mostly concentrated on protocols with *closed-ended data structures*, where messages exchanged between principals have fixed and finite format. In what follows, we will refer to these protocols as *closed-ended protocols*. In many protocols, however, the data structures are *open-ended* (see Section 2 for examples): the number of data fields that must be processed by a principal in one receive-send action is unbounded, where receive-send action means that a principal receives a message and reacts, after some internal computation, by sending a message. We will call protocols with open-ended data structures *open-ended*.

This paper addresses open-ended protocols, and thus, deals with one of the challenges pointed out by Meadows [16]. The goal is to devise a protocol model rich enough to capture a large class of open-ended protocols such that security is decidable; the long-term goal is to develop tools for automatic verification of open-ended protocols.

Open-ended protocols make it necessary to model principals who can perform in one receive-send action an unbounded number of *internal actions*; only then can they handle open-ended data structures. Therefore, the first problem is to find a good computational model for receive-send actions. It turns out that one cannot simply extend the existing models. More specifically, Rusinowitch and Turuani [20] describe receive-send actions by single rewrite rules and show security to be NP-complete. In this model, principals have unbounded memory. Furthermore, the terms in the rewrite rules may be non-linear, i.e., multiple occurrence of one variable is allowed, and thus, a principal can compare messages of arbitrary size for equality. To handle open-ended protocols, we generalize the model by Rusinowitch and Turuani in a canonical way and show that if receive-send actions are described by *sets* of rewrite rules, security is undecidable, even with i) finite memory and non-linear terms, or ii) unbounded memory and linear terms. Consequently, we need a computational model in which principals have finite memory and cannot compare messages of arbitrary size for equality.

For this reason, we propose to use transducers, i.e., finite automata with output, as the computational model for receive-send actions, since transducers satisfy the above restrictions — they have finite memory and cannot compare messages of arbitrary size for equality —, and still can deal with open-ended data structures. In Section 5.1 our so-called transducer-based model is discussed in detail. The main technical result of this paper is that in the transducer-based model, security is decidable and PSPACE-hard under the following assumptions: the number of sessions is bounded, i.e., a protocol is analyzed assuming a fixed number of interleaved protocol runs; nonces and complex keys are not allowed. We, however, put no restrictions on the Dolev-Yao intruder; in particular, the message size is unbounded. These are standard assumptions also made in most decidable models for closed-ended protocols [20, 2, 13].[1] Just as in these works, the security property we study in the present paper is secrecy.

The results indicate that from a computational point of view, the analysis of open-ended protocols is harder than for closed-ended protocols, for which security is "only" NP-complete [20]. The additional complexity comes from the fact that now we have, beside the Dolev-Yao intruder, another source of infinite behavior: the unbounded number of internal actions (i.e., paths in the transducers of unbounded length). This makes it necessary to devise new proof techniques to show decidability. Roughly speaking, using that transducers only have finite memory we will use a pumping argument showing that the length of paths in the transducers can be bounded in the size of the problem instance.

Related work. All decidability and undecidability results obtained so far only apply to closed-ended protocols. Decidability depends on the following parameters: bounded or unbounded number of sessions, bounded or unbounded message size, absence or presence of pairing, nonces, and/or complex keys.

Usually, if one allows for an unbounded number of sessions, security is undecidable [1, 8, 2, 9], except when the message size is bounded and nonces are

[1] In [20, 13], however, complex keys are allowed.

disallowed [8] (then security is EXPTIME-complete), or pairing is disallowed [6, 2] (then security is in P). The situation is much better if one puts a bound on the number of sessions and disallows nonces. Then, even with pairing and unbounded message size, security is decidable [20, 2, 13] and NP-hard [20, 2]; Rusinowitch and Turuani [20] show NP-completeness (with complex keys). Except for complex keys, we make exactly the same assumptions in our models.

To the best of our knowledge, the only contributions on formal analysis of open-ended protocols are the following: The recursive authentication protocol [5] has been analyzed by Paulson [18], using the Isabelle theorem prover, as well as by Bryans and Schneider [4], using the PVS theorem prover; the A-GDH.2 protocol [3] has been analyzed by Meadows [15] with the NRL Analyzer, and manually by Pereira and Quisquater [19], based on a model similar to the strand spaces model. As mentioned, decidability issues have not been studied so far.

Structure of the paper. In Section 2, we give examples of open-ended protocols. We then define a generic model for describing open-ended protocols (Section 3). In this model, receive-send actions can be arbitrary computations. In Section 4, we consider the instances of the generic model in which receive-send actions are specified by sets of rewrite rules, and show the mentioned undecidability result. The transducer-based model, the instance of the generic protocol model in which receive-send actions are given by transducers, is introduced in Section 5. This section also contains the mentioned discussion. In Section 6 the actual decidability and complexity results are stated. Finally, we conclude in Section 7.

Due to space limitations, in this paper we have largely omitted technical details, and rather focused on the introduction and the discussion of our transducer-based model. The detailed and often involved proofs of all our results can be found in the technical report [14]. It also shows how the recursive authentication protocol (see Section 2) can be described in our transducer-based model.

2 Examples of Open-ended Protocols

An example of an open-ended protocol is the IKE Protocol [12], in which a principal needs to pick a *security association (SA)*, the collection of algorithms and other informations used for encryption and authentication, among an a priori unbounded list of SAs. Such a list is an open-ended data structure, since it has an unbounded number of data fields to be examined by a principal. An attack on IKE, found by Zhou [22] and independently Ferguson and Schneier [10], shows that when modeling open-ended protocols, the open-ended data structures must be taken into account, since otherwise some attacks might not be found. In other words, as also pointed out by Meadows [16], open-endedness is security relevant.

Other typical open-ended protocols are group protocols, for example, the recursive authentication protocol (RA protocol) [5] and the A-GDH.2 protocol [3], which is part of the CLIQUES project [21]. In the RA protocol, a key distribution server receives an a priori unbounded sequence of request messages (containing pairs of principals who want to share session keys) and must generate a corresponding sequence of certificates (containing the session keys). These sequences are open-ended data structures: In one receive-send action the server needs to process an unbounded number of data fields, namely the sequence of pairs of principals. Group protocols often allow for an unbounded number of receive-send actions in one protocol run. In our models, we will, however, always assume a fixed bound on the number of receive-send actions, since otherwise, just as in the case of an unbounded number of sessions, security in general leads to undecidability. Nevertheless, even with such a fixed bound it is still necessary to model open-ended data structures. In the RA protocol, a bound on the number of receive-send actions would imply that the sequence of requests generated by the principals is bounded. Nevertheless, the intruder can generate arbitrarily long request messages. Thus, the data structures are still open-ended, and the server should be modeled in such a way that, as in the actual protocol, he can process open-ended data structures.

In [14], we provide a formal description of the RA protocol in our transducer-based model.

3 A Generic Protocol Model

Our generic protocol model and the underlying assumptions basically coincide with the ones proposed by Rusinowitch et al. [20] and Amadio el al. [2] for closed-ended protocols. However, the important difference is that in the generic model, receive-send actions are, roughly speaking, binary relations over the message space, and thus can be interpreted as arbitrary computations. In the models of Rusinowitch el al. and Amadio et al. , receive-send actions are described by single rewrite rules or processes without loops, respectively.

Thus, the generic protocol model is a very general framework for open-ended protocols. In fact, it is much too general to study decidability issues. Therefore, in subsequent sections we will consider different instances of this model.

The main features of the generic protocol model can be summarizes as follows:

- a generic protocol is described by a finite set of principals;
- the internal state space of a principal may be infinite (which, for example, enables a principal to store arbitrarily long messages);
- every principal is described by a finite sequence of receive-send actions;
- receive-send actions are arbitrary computations.

We make the following assumptions:

- the intruder is the standard Dolev-Yao intruder; in particular, we do not put restrictions on the size of messages;
- principals and the intruder cannot generate nonces;
- keys are atomic;
- the number of sessions is bounded. More precisely, the sessions considered in the analysis are only those encoded in the protocol description itself.

These are standard assumptions also made in decidable models for closed-ended protocols. They coincide with the ones in [2], and except for complex keys, with those in [20, 13].

Let us now give a formal definition of the generic protocol model.

3.1 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. The *set of messages* (over $\mathcal{N}$) is the least set $\mathcal{M}$ that satisfies the following properties:

- $\mathcal{N} \subseteq \mathcal{M}$;
- if $m, m' \in \mathcal{M}$, then $mm' \in \mathcal{M}$;
- if $m \in \mathcal{M}$ and $a \in \mathcal{N}$, then $\mathtt{enc}_a(m) \in \mathcal{M}$;
- if $m \in \mathcal{M}$, then $\mathtt{hash}(m) \in \mathcal{M}$.

As usual, concatenation is an associative operation, i.e., $(mm')m'' = m(m'm'')$. Note that we only allow for atomic keys, i.e., in a message $\mathtt{enc}_a(\cdot)$, a is always an atomic message.

Let ε denote the *empty message* and $\mathcal{M}_\varepsilon := \mathcal{M} \cup \{\varepsilon\}$ the set of messages containing ε. Note that ε is not allowed inside encryptions or hashes, that is, $\mathtt{enc}_a() \notin \mathcal{M}_\varepsilon$ and $\mathtt{hash}() \notin \mathcal{M}_\varepsilon$.

Later, we will also consider terms, i.e., messages with variables. Let $V := \{v_0, \ldots, v_{n-1}\}$ be a set of variables. Then a *term* t (over V) is a message over the atomic messages $\mathcal{N} \cup V$, where variables are not allowed as keys, i.e., terms of the form $\mathtt{enc}_v(\cdot)$ for some variable v are forbidden. A *substitution* σ is a mapping from V into $\mathcal{M}_\varepsilon$. If t is a term, then $\sigma(t)$ denotes the message obtained from t by replacing every variable v in t by $\sigma(v)$.

The *depth* $\mathsf{depth}(t)$ *of a term* t is the maximum number of nested encryptions and hashes in t, i.e.,

- $\mathsf{depth}(\varepsilon) := 0$, $\mathsf{depth}(a) := 0$ for every $a \in \mathcal{N} \cup V$;
- $\mathsf{depth}(tt') := \mathsf{max}\{\mathsf{depth}(t), \mathsf{depth}(t')\}$;
- $\mathsf{depth}(\mathtt{enc}_a(t)) := \mathsf{depth}(t) + 1$; $\mathsf{depth}(\mathtt{hash}(t)) := \mathsf{depth}(t) + 1$.

3.2 The Intruder Model

We use the standard Dolev-Yao intruder model [7]. 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. As usual in the Dolev-Yao model, we make the perfect cryptography assumption. 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 $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$ is the smallest set satisfying the following conditions:

- $\mathcal{K} \subseteq \mathsf{d}(\mathcal{K})$;

– if $mm' \in \mathsf{d}(\mathcal{K})$, then $m \in \mathsf{d}(\mathcal{K})$ and $m' \in \mathsf{d}(\mathcal{K})$ (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 $m \in \mathsf{d}(\mathcal{K})$ and $m' \in \mathsf{d}(\mathcal{K})$, then $mm' \in \mathsf{d}(\mathcal{K})$ (composition);
– if $m \in \mathsf{d}(\mathcal{K})$, $m \neq \varepsilon$, and $a \in \mathcal{N} \cap \mathsf{d}(\mathcal{K})$, then $\mathtt{enc}_a(m) \in \mathsf{d}(\mathcal{K})$ (encryption);
– if $m \in \mathsf{d}(\mathcal{K})$ and $m \neq \varepsilon$, then $\mathtt{hash}(m) \in \mathsf{d}(\mathcal{K})$ (hashing).

3.3 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. Since we are interested in attacks, the definition of a protocol also contains the initial intruder knowledge. Formally, principals and protocols are defined as follows.

Definition 1. *A* generic principal Π *is a tuple* (Q, I, n, α) *where*

– *Q is the (possibly infinite) set of states of Π;*
– *I is the set of initial states of Π;*
– *n is the number of receive-send actions to be performed by Π;*
– *α is a mapping assigning to every $j \in \{0, \dots, n-1\}$ a receive-send action $\alpha(j) \subseteq Q \times \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon \times Q$.*

A generic protocol P *is a tuple* $(n, \{\Pi_i\}_{i<n}, \mathcal{K})$ *where*

– *n is the number of principals;*
– *$\{\Pi_i\}_{i<n}$ is a family of n generic principals, and*
– *$\mathcal{K} \subseteq \mathcal{M}_\varepsilon$ is the initial intruder knowledge.*

Note that receive-send actions are arbitrary relations. Intuitively, they take an input message (2. component) and nondeterministically, depending on the current state (1. component), return an output message (3. component) plus a new state (4. component). Later, when we consider instances of the generic protocol model, one receive-send action of a principal will consist of an unbounded number of internal actions. By allowing receive-send actions to be nondeterministic and principals to have a set of initial states, instead of a single initial state, one can model more flexible principals: for instance, those that nondeterministically choose one principal, who they want to talk to, or one SA from the list of SAs in the IKE Protocol.

We also remark that a protocol P is *not* parametrized by n. In particular, when we say that P is secure, we mean that P is secure given the n principals as defined in the protocol. We do not mean that P is secure for every number n of principals.

3.4 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 2. *Let $P = (n, \{\Pi_i\}_{i<n}, \mathcal{K})$ be a generic protocol with $\Pi_i = (Q_i, I_i, n_i, \alpha_i)$, 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);*[2]
- *a mapping ψ assigning to every (i,j), $i < n$, $j < n_i$, a tuple*
$$\psi(i,j) = (q_i^j, m_i^j, {m'}_i^j, q_i^{j+1})$$
with
  - $q_i^j, q_i^{j+1} \in Q_i$ *(the state of Π_i before/after performing $\alpha_i(j)$); and*
  - $m_i^j, {m'}_i^j \in \mathcal{M}_\varepsilon$ *(the input message received and output message sent by $\alpha_i(j)$);*

such that

- $q_i^0 \in I_i$ *for every $i < n$;*
- $m_i^j \in \mathsf{d}(\mathcal{K} \cup \{{m'}_{i'}^{j'} \mid (i',j') < (i,j)\})$ *for every $i < n$, $j < n_i$;*
- $(q_i^j, m_i^j, {m'}_i^j, q_i^{j+1}) \in \alpha_i(j)$ *for every $i < n$, $j < n_i$.*

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

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*.

Whether ATTACK is decidable or not heavily depends on what kinds of receive-send actions a principal is allowed to perform. In the subsequent sections, we look at different instances of generic protocols, i.e., different computational models for receive-send actions, and study the problem ATTACK for the classes of protocols thus obtained.

4 Undecidability Results

We extend the model proposed by Rusinowitch and Turuani [20] in a straightforward way such that open-ended protocols can be handled, and show that this extension leads to undecidability of security.

The model by Rusinowitch and Turuani can be considered as the instance of the generic protocol model in which receive-send actions are described by single rewrite rules of the form $t \to t'$, where t and t' are terms.[3] The internal state of

[2] Although, we assume a linear ordering on the receive-send actions performed by a principal, we could as well allow partial orderings (as in [20]) without any impact on the decidability and complexity results.

[3] Since Rusinowitch and Turuani allow for complex keys, the terms are more general than the ones we use here. However, we will only consider terms as defined in Section 3.1.

a principal is given implicitly by the values assigned to the variables occurring in the rewrite rules – different rules may share variables. In particular, a principal has unbounded memory to store information for use in subsequent receive-send actions. Roughly speaking, a message m is transformed by a receive-send action of the form $t \to t'$ into the message $\sigma(t')$, where σ is a substitution satisfying $m = \sigma(t)$. In [20], it is shown that in this setting, ATTACK is NP-complete.

Of course, in this model open-ended data structures cannot be handled since the left hand-side of a rewrite rule t has a fixed and finite format, and thus, one can only process messages with a fixed number of data fields.

A natural extension of this model, which allows to deal with open-ended data structures, is to describe receive-send actions by sets of rewrite rules, which can nondeterministically be applied to the input message, where, as in the model of Rusinowitch and Turuani, rewriting means top-level rewriting: If the rule $t \to t'$ is applied to the input message m yielding $\sigma(t')$ as output, another rule (nondeterministically chosen from the set of rules) may be applied to $\sigma(t')$. To the resulting output yet another rule may be applied and so on, until no rule is or can be applied anymore. The applications of the rules are the internal actions of principals. The instance of the generic protocol model in which receive-send actions are described by sets of rewrite rules as described above is called *rule-based protocol model*. In [14], we give a formal definition of this model. In this model, we distinguish between input, output, and process rules, and also put further restrictions on the rewrite rules such that they can be applied only a finite (but a priori unbounded) number of times.

Theorem 1. *For rule-based protocols,* ATTACK *is undecidable.*

By reduction from Post's Correspondence Problem (PCP), this theorem is easy to show. It holds true, even for protocols consisting of only one principal, which may only perform one receive-send action. In other words, the undecidability comes from the internal actions alone.

However, the reduction does not work if only linear terms are allowed in rewrite rules. In linear terms, every variable occurs at most once, and therefore, one cannot compare submessages of arbitrary size for equality. Nevertheless, if principals can store one message and compare it with a submessage of the message being processed, we still have undecidability. Such protocols are called *linear-term one-memory protocols*; see [14] for the formal definition and the proof of undecidability, which is again by a rather straightforward reduction from PCP.

Theorem 2. *For linear-term one-memory protocols,* ATTACK *is undecidable.*

5 The Transducer-based Protocol Model

The previous section indicates that, informally speaking, when principals can process open-ended data structures and, in addition, can

1. compare submessages of arbitrary size (which is possible if terms are not linear), or

2. store one message and compare it with a submessage of the message being processed,

then security is undecidable. To obtain decidability, we need a device with only finite memory, and which does not allow to compare messages of arbitrary size. This motivates to use transducers to describe receive-send actions. In what follows, we define the corresponding instance of the generic protocol model. In Section 5.1, we will discuss capabilities and restrictions of our transducer-based model.

If Σ is a finite alphabet, Σ^* will denote the set of finite words over Σ, including the empty word ε.

Definition 3. *A* transducer $\mathcal{A}$ *is a tuple* $(Q, \Sigma, \Omega, I, \Delta, F)$ *where*

- Q *is the finite set of states of* $\mathcal{A}$*;*
- Σ *is the finite input alphabet;*
- Ω *is the finite output alphabet;*
- $I \subseteq Q$ *is the set of initial states of* $\mathcal{A}$*;*
- $\Delta \subseteq Q \times \Sigma^* \times \Omega^* \times Q$ *is the finite set of transitions of* $\mathcal{A}$*; and*
- $F \subseteq Q$ *is the set of final states of* $\mathcal{A}$*.*

A *path* π (of length n) in $\mathcal{A}$ from p to q is of the form $q_0(v_0, w_0)q_1(v_1, w_1)q_2 \dots (v_{n-1}, w_{n-1})q_n$ with $q_0 = p$, $q_n = q$, and $(q_i, v_i, w_i, q_{i+1}) \in \Delta$ for every $i < n$; π is called *strict* if $n > 0$, and v_0 and v_{n-1} are non-empty words. The word $v_0 \dots v_{n-1}$ is the *input label* and $w_0 \dots w_{n-1}$ is the *output label* of π. A path of length 0 has input and output label ε. We write $p(v, w)q \in \mathcal{A}$ ($p(v, w)q \in_s \mathcal{A}$) if there exists a (strict) path from p to q in $\mathcal{A}$ with input label v and output label w.

If $S, T \subseteq Q$, then $\mathcal{A}(S, T) := \{(p, v, w, q) \mid p \in S, q \in T, p(v, w)q \in \mathcal{A}\} \subseteq Q \times \Sigma^* \times \Omega^* \times Q$. The *output of* $\mathcal{A}$ *on input* $v \in \Sigma^*$ is defined by $\mathcal{A}(v) := \{w \mid$ there exists $p \in I$ and $q \in F$ with $(p, v, w, q) \in \mathcal{A}(I, F)\}$.

If $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Omega \cup \{\varepsilon\}) \times Q$, $\mathcal{A}$ is called *transducer with letter transitions* in contrast to transducers with word transitions. The following remark shows that it suffices to consider transducers with letter transitions.

Remark 1. Let $\mathcal{A} = (Q, \Sigma, \Omega, I, \Delta, F)$ be a transducer. Then there exists a transducer $\mathcal{A}' = (Q', \Sigma, \Omega, I, \Delta', F)$ with letter transitions such that $Q \subseteq Q'$, and $\mathcal{A}'(S, T) = \mathcal{A}(S, T)$ for every $S, T \subseteq Q$.

In order to specify the receive-send actions of a principal, we consider special transducers, so-called message transducers, which satisfy certain properties. Message transducers interpret messages as words over the finite alphabet $\Sigma_{\mathcal{N}}$, consisting of the atomic messages as well as the letters "$\mathtt{enc}_a($", "$\mathtt{hash}($", and "$)$", that is,

$$\Sigma_{\mathcal{N}} := \mathcal{N} \cup \{\mathtt{enc}_a(\mid a \in \mathcal{N}\} \cup \{\mathtt{hash}(,)\}.$$

Messages considered as words over $\Sigma_{\mathcal{N}}$ have always a balanced number of opening parentheses, i.e., "$\mathtt{enc}_a($" and "$\mathtt{hash}($", and closing parentheses, i.e., "$)$".

A message transducer reads a message (interpreted as a word) from left to right, thereby producing some output. If messages are considered as finite trees (where leaves are labeled with atomic messages and internal nodes are labeled with the encryption or hash symbol), a message transducer traverses such a tree from top to bottom and from left to right.

Definition 4. *A* message transducer $\mathcal{A}$ *(over* $\mathcal{N}$*) is a tuple* $(Q, \Sigma_{\mathcal{N}}, I, \Delta, F)$ *such that* $(Q, \Sigma_{\mathcal{N}}, \Sigma_{\mathcal{N}}, I, \Delta, F)$ *is a transducer with letter transitions, and*

1. *for every* $x \in \mathcal{M}_\varepsilon$*,* $\mathcal{A}(x) \subseteq \mathcal{M}_\varepsilon$*; and*
2. *for all* $p, q \in Q$*,* $x \in \mathcal{M}$*, and* $y \in \Sigma_{\mathcal{N}}^*$*, if* $p(x,y)q \in_s \mathcal{A}$*, then* $y \in \mathcal{M}_\varepsilon$*.*

The first property is a condition on the "external behavior" of a message transducer: Whenever a message transducer gets a message as input, then the corresponding outputs are also messages (rather than arbitrary words). Note that in an attack, the input to a message transducer is always a message. The second property imposes some restriction on the "internal behavior" of a message transducer. Both properties do not seem to be too restrictive. They should be satisfied for most protocols; at least they are for the transducers in the model of the recursive authentication protocol (as described in [14]).

An open issue is whether the properties on the internal and external behavior are decidable, i.e., given a transducer over $\Sigma_{\mathcal{N}}$ does it satisfy 1. and 2. of Definition 4. Nevertheless, in the model of the recursive authentication protocol it is easy to see that the transducers constructed satisfy the properties.

For $S, T \subseteq Q$, we define $M_{\mathcal{A}}(S,T) := \mathcal{A}(S,T) \cap (Q \times \mathcal{M}_\varepsilon \times \Sigma_{\mathcal{N}}^* \times Q)$. By the definition of message transducers, $M_{\mathcal{A}}(I,F) \subseteq (Q \times \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon \times Q)$ if I is the set of initial states and F is the set of final states of $\mathcal{A}$. Thus, message transducers specify receive-send actions of principals (in the sense of Definition 1) in a natural way.

In order to define one principal (i.e., the whole sequence of receive-send actions a principal performs) by a single transducer, we consider so-called extended message transducers: $\mathcal{A} = (Q, \Sigma_{\mathcal{N}}, \Delta, (I_0, \dots, I_n))$ is an *extended message-transducer* if $\mathcal{A}_{I_j, I_{j+1}} := (Q, \Sigma_{\mathcal{N}}, I_j, \Delta, I_{j+1})$ is a message transducer for all $j < n$. Given such an extended message transducer, it defines the principal (Q, I_0, n, α) with $\alpha(j) = M_{\mathcal{A}_{I_j, I_{j+1}}}(I_j, I_{j+1})$ for $j < n$. In this setting, an internal action of a principal corresponds to applying one transition in the extended message transducer.

Definition 5. *A* transducer-based protocol P *is a generic protocol where the principals are defined by extended message transducers.*

5.1 Discussion of the Transducer-based Protocol Model

In this section, we aim at clarifying capabilities and limitations of the transducer-based protocol model. To this end, we compare this model with the models usually used for closed-ended protocols. To make the discussion more concrete, we concentrate on the model proposed by Rusinowitch and Turuani (see Section 4), which, among the decidable models used for closed-ended protocols, is very powerful. In what follows, we refer to their model as the *rewriting model*. As pointed out in Section 3, the main difference between the two models is the way receive-send actions are described. In the rewriting model receive-send actions are described by single rewrite rules and in the transducer-based model by message transducers.

Let us start to explain the capabilities of message transducers compared to rewrite rules.

Open-ended data structures. As mentioned in Section 4, with a single rewrite rule one cannot process an unbounded number of data fields. This is, however, possible with transducers.

For example, considering the IKE protocol (see Section 2), it is easy to specify a transducer which i) reads a list of SAs, each given as a sequence of atomic messages, ii) picks one SA, and iii) returns it. With a single rewrite rule, one could not parse the whole list of SAs.

The transducer-based model of the recursive authentication protocol (described in [14]) shows that transducers can also handle more involved open-ended data structures: The server in this protocol generates a sequence of certificates (containing session keys) from a *request message* of the form $\texttt{hash}(m_0\texttt{hash}(m_1 \cdots \texttt{hash}(m_n)\cdots)$, where the m_i's are sequences of atomic messages and the nesting depth of the hashes is a priori unbounded (see [14] for the exact definition of the messages.)

Of course, a transducer cannot match opening and closing parenthesis, if they are nested arbitrarily deep, since messages are interpreted as words. However, often this is not necessary: In the IKE protocol, the list of SAs is a message without any nesting. In the recursive authentication protocol, the structure of request messages is very simple, and can be parsed by a transducer. Note that a transducer does not need to check whether the number of closing parenthesis in the request message matches the number of hashes because all words sent to a message transducer (by the intruder) are messages, and thus, well-formed.

Simulating rewrite rules. Transducers can simulate certain receive-send actions described by single rewrite rules. Consider for example the rule $\texttt{enc}_k(x) \rightarrow \texttt{hash}(kx)$, where x is a variable and k an atomic message: First, the transducer would read "$\texttt{enc}_k($" and output "$\texttt{hash}(k$", and then read, letter by letter, the rest of the input message, i.e., "$x)$" – more precisely, the message substituted for x – and simultaneously write it into the output.

Let us now turn to the limitations of the transducer-based model compared to the rewriting model. The main limitations are the following:

1. *Finite memory:* In the rewriting model, principals can store messages of arbitrary size to use them in subsequent receive-send actions. This is not possible with transducers, since they only have finite memory.

2. *Comparing messages:* In the rewriting model, principals can check whether submessages of the input message coincide. For example, if $t = \mathtt{hash}(kx)x$, with k an atomic message and x a variable, a principal can check whether plain text and hash match. Transducers cannot do this.
3. *Copying messages:* In the rewriting model, principals can copy messages of arbitrary size. For example, in the rule $\mathtt{enc}_k(x) \to \mathtt{hash}(kx)x$, the message x is copied. Again, a transducer would need to store x in some way, which is not possible because of the finite memory. As illustrate above, a transducer could however simulate a rule such as $\mathtt{enc}_k(x) \to \mathtt{hash}(kx)$.
4. *Linear terms:* A transducer cannot simulate all rewrite rules with linear left and right hand-side. Consider for example the rule $\mathtt{enc}_k(xAy) \to \mathtt{hash}(yAx)$, where x and y are variables, and A is an atomic message. Since in the output, the order of x and y is switched, a transducer would have to store the messages substituted for x and y. However, this requires unbounded memory.

The undecidability results presented in Section 4 indicate that, if open-ended data structures are involved, the restrictions 1. and 2. seem to be unavoidable. The question is whether this is also the case for the remaining two restrictions. We will comment on this below.

First, let us point out some work-arounds. In 1., it often (at least under reasonable assumptions) suffices to store atomic messages such as principal names, keys, and nonces. Thus, one does not always need unbounded memory. One example is the recursive authentication protocol. In 4., it might be possible to modify the linear terms such that they can be parsed by a message transducer, and such that the security of the protocol is not affected. In the example, if one changes the order of x and y in the output, the rewrite rule can easily be simulated by a transducer. Finally, a work-around for the restrictions 2. to 4., is to put a bound on the size of messages that can be substituted for the variables. This approach is usually pursued in protocol analysis based on finite-state model checking (e.g., [17]), where transducers have the additional advantage of being able to process open-ended data structures. For messages of bounded size, all transformations performed by rewrite rules can also be carried out by message transducers. Moreover, in this setting message transducers can handle type flaws.

Of course, it is desirable to avoid such work-arounds if possible to make the analysis of a protocol more precise and reliable. One approach, which might lift some of the restrictions (e.g., 3. and 4.), is to consider tree transducers instead of word transducers to describe receive-send actions. It seems, however, necessary to devise new kinds of tree transducers or extend existing once, for example tree transducers with look-ahead, that are especially tailored to modeling receive-send actions. A second approach is to combine different computational models for receive-send actions. For instance, a hybrid model in which some receive-actions are described by rewrite rules and others by transducers might still be decidable.

6 The Main Result

The main technical result of this paper is the following:

Theorem 3. *For transducer-based protocols,* ATTACK *is decidable and PSPACE-hard.*

In what follows, we very briefly sketch the proof idea of the theorem. See [14] for the detailed proof.

The hardness result is easy to show. It is by reduction from the finite automata intersection problem, which has been shown to be PSPACE-complete by Kozen [11].

The decidability result is much more involved, because we have two sources of infinite behavior in the model. First, the intruder can perform an unbounded number of steps to derive a new message, and second, to perform one receive-send action, a principal can carry out an unbounded number of internal actions. Note that because transducers may have ε-transitions, i.e., not in every transition a letter is read from the input, the number of transitions taken in one receive-send action is not even bounded in the size of the input message or the problem instance.

While the former source of infinity was already present in the (decidable) models for closed-ended protocols [20, 2, 13], the latter is new. To show Theorem 3, one therefore not only needs to show that the number of actions performed by the intruder can be bounded, but also the number of internal actions of principals. In fact, it suffices to show the latter, since if we can bound the number of internal actions, a principal only reads messages of bounded length and therefore the intruder only needs to produce messages of size bounded by this length.

Roughly speaking, we can bound the number of internal actions of principals because principals (the extended message transducers describing them) have only finite memory and we can apply a pumping argument showing that long paths in a message transducer can be truncated. Of course, the pumping argument is much more involved than that for usual finite automata since we have to guarantee that i) the input label of the truncated path is still a message that can be produced by the intruder, and ii) the output label of the truncated path is such that it can be used by the intruder to perform a successful attack if this was possible with the output label of the original path. While property i) is relatively easy to achieve, property ii) is very involved. In what follows, we therefore concentrate on ii).

Proof idea of the pumping argument. First, we define a so-called *solvability preserving (quasi-)ordering* on messages, which allows to replace single messages in the intruder knowledge by new ones such that if in the original problem a successful attack exists, then also in the modified problem. This reduces the pumping argument to the following problem: Truncate paths in message transducers in such a way that the output of the original path is equivalent (w.r.t. the solvability preserving ordering) to the output of the truncated path. It remains to find criteria for truncating paths in this way. To this purpose, we introduce another quasi-ordering, the so-called *path truncation ordering*, which indicates at which positions a path can be truncated. To really obtain a bound on the length of paths, one then needs to show that the equivalence relation corresponding to the path truncation ordering has finite index – more accurately, an index that can be bounded in the size of the problem instance. With this, and the fact that message transducers have only finite memory, the length of paths can be restricted. Finally, to show the bound on the index, one needs to establish a bound on the depth of messages (i.e., the depth of nested encryptions and hashes) in successful attacks. Again, we make use of the fact that message transducers have only finite memory.

7 Conclusion

We have introduced a generic protocol model for analyzing the security of open-ended protocols, i.e., protocols with open-ended data structures, and investigated the decidability of different instances of this model. In one instance, receive-send actions are modeled by sets of rewrite rules. We have shown that in this instance, security is undecidable. These results indicated that to obtain decidability, principals should only have finite memory and should not be able to compare messages of arbitrary size. This motivated our transducer-based model, which complies to these restrictions, but still captures certain open-ended protocols. We have shown that in this model security is decidable and PSPACE-hard; it remains to establish a tight complexity bound. These results have been shown for the shared key setting and secrecy properties. We conjecture that they carry over rather easily to public key encryption and authentication.

As pointed out in Section 5.1, a promising future direction is to combine the transducer-based model with the models for closed-ended protocols and to devise tree transducers suitable for describing receive-send actions. We will also try to incorporate complex keys, since they are used in many protocols. We believe that the proof techniques devised in this paper will help to show decidability also in the more powerful models. Finally, encouraged by the work that has been done for closed-ended protocols, the long-term goal of the work started here is to develop tools for automatic verification of open-ended protocols, if possible by integrating the new algorithms into existing tools.

Acknowledgement I thank Thomas Wilke for many helpful comments on this work, and Catherine Meadows for pointing me to the paper by Pereira and Quisquater.

References

1. R.M. Amadio and W. Charatonik. On name generation and set-based analysis in Dolev-Yao model. Technical Report RR-4379, INRIA, 2002.
2. R.M. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. Technical Report RR-4147, INRIA, 2001.

3. G. Ateniese, M. Steiner, and G. Tsudik. Authenticated group key agreement and friends. In *Proceedings of the 5th ACM Conference on Computer and Communication Secruity (CCS'98)*, pages 17–26, San Francisco, CA, 1998. ACM Press.
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. D. Dolev, S. Even, and R.M. Karp. On the Security of Ping-Pong Protocols. *Information and Control*, 55:57–68, 1982.
7. D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
8. N.A. Durgin, P.D. Lincoln, J.C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.
9. S. Even and O. Goldreich. On the Security of Multi-Party Ping-Pong Protocols. In *IEEE Symposium on Foundations of Computer Science (FOCS'83)*, pages 34–39, 1983.
10. N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. Technical report, 2000. Available from http://www.counterpane.com/ipsec.pdf.
11. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
12. D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*, November 1998. RFC 2409.
13. A. Huima. Efficient infinite-state analysis of security protocols. In *Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.
14. R. Küsters. On the Decidability of Cryptographic Protocols with Open-ended Data Structures. Technical Report IFI-0204, CAU Kiel, Germany, 2002. Available from http://www.informatik.uni-kiel.de/reports/2002/0204.html.
15. C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In *Proceedings of the First Workshop on Issues in the Theory of Security (WITS'00)*, pages 87–92, 2000.
16. C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proc. of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2000.
17. J. Mitchell, M. Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols using Murphi. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society Press, 1997.
18. L.C. Pauslon. Mechanized proofs for a recursive authetication protocol. In *10th IEEE Computer Security Foundations Workshop (CSFW-10)*, pages 84–95, 1997.
19. 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.
20. 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.
21. M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to key agreement. In *IEEE International Conference on Distributed Computing Systems*. IEEE Computer Society Press, pages 380–387, 1998.
22. J. Zhou. Fixing a security flaw in IKE protocols. *Electronic Letter*, 35(13):1072–1073, 1999.