INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

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

Ralf Küsters

Bericht Nr. 0204
February 2002

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

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

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

Ralf Küsters

Bericht Nr. 0204
February 2002

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

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

Ralf Küsters
Institut für Informatik und Praktische Mathematik
CAU 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 [15] 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 [15]. 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 they can handle open-ended data structures. Thus, 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. The rewrite rules may contain non-linear terms (in particular, messages can be checked for equality) and the principals have unbounded memory. To handle open-ended protocols, we generalize their model 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. Thus, 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, and still, can deal with open-ended data structures. Section 5.1 contains a detailed discussion on our so-called transducer-based model. 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; and 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]

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 also 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: 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 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

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

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 [14] 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 two 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. Section 6 contains the actual decidability result and in Section 7 we prove the complexity lower bound. Finally, we conclude in Section 8. The appendix provides a detailed description of the recursive authentication protocol (Appendix A) and a formalization of this protocol in the transducer-based protocol model (Appendix B).

2 Examples of Open-ended Protocols

One example of an open-ended protocol is the IKE Protocol [12], in which a principal needs to pick a *security association (SA)* among an a priori unbounded list of SAs, where an SA is the collection of algorithms and other informations used for encryption and authentication. Such a list of SAs 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 [15], open-endedness is security relevant.

Other typical open-ended protocols are group protocols, for example, the recursive authentication protocol [5] and the A-GDH.2 protocol [3], which is part of the CLIQUES project [21]. Group protocols often allow for an unbounded number of receive-send actions in one protocol run. In our models, we will, however, 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. The most important reason is that the intruder can produce messages with an unbounded number of data fields, which must be processed by the principals.

Also, as in the IKE protocol, open-ended data structures may occur independently of whether there is a bound on the number of receive-send actions or not.

In Appendix A, the recursive authentication protocol is described in detail, and in Appendix B we provide a formalization in the transducer-based protocol 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 to general to study decidability issues. In the subsequent sections we will consider different instances of this model.

The main features of the generic protocol model can be summarized 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; in particular, 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 $\mathtt{hash}() \notin \mathcal{M}_\varepsilon$.

The *size* $|m|$ of a message m is the number of occurrences of atomic messages, encryption, and hash functions in m; $|\varepsilon| := 0$.

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) (also written $t(v_0, \ldots, v_{n-1})$) 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. Let $\mathcal{T}(V)$ denote the set of terms over V and $\mathcal{T}_\varepsilon(V) := \mathcal{T}(V) \cup \{\varepsilon\}$. For $t_0, \ldots, t_{n-1}, t \in \mathcal{T}_\varepsilon(V)$, $t[v_0/t_0, \ldots, v_{n-1}/t_{n-1}]$ denotes the term obtained from t by simultaneously substituting the variables v_i by t_i. A term $t' \in \mathcal{T}_\varepsilon(V)$ is a *subterm* of $t \in \mathcal{T}_\varepsilon(V)$ if there exists a term $t'' \in \mathcal{T}(V \cup \{v\})$, where v is a new variable occurring exactly once in t'', such that $t = t''[v/t']$. A *substitution* σ is a mapping from V into $\mathcal{M}_\varepsilon$. If $t \in \mathcal{T}_\varepsilon(V)$, 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 $a \in \mathcal{N} \cup V$;
- $\mathsf{depth}(tt') := \max\{\mathsf{depth}(t), \mathsf{depth}(t')\}$;
- $\mathsf{depth}(\mathtt{enc}_a(t)) := \mathsf{depth}(t) + 1$; $\mathsf{depth}(\mathtt{hash}(t)) := \mathsf{depth}(t) + 1$.

Given a finite subset of messages $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$, the maximum depth of messages in $\mathcal{K}$ is denoted by

$$\mathsf{depth}(\mathcal{K}) := \max\{\mathsf{depth}(m) \mid m \in \mathsf{d}(\mathcal{K})\}.$$

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).

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. 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. Formally this means (see, e.g., [2] for a proof):

Lemma 1 *If $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$, then $\mathsf{d}(\mathcal{K}) = \mathsf{syn}(\mathsf{an}(\mathcal{K}))$.*

3.3 Protocols

Protocols are described by sets of principals and every principal is defined by a sequence of receive-send actions, which 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 2 *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 $j \in \{0, \ldots, n-1\}$ 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 and nondeterministically, depending on the current state, return an output message plus a new state. 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.

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 3 *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

[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.

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 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 non-deterministically be applied to the input message. More precisely, we will consider two different extensions: In the first extension, principals do not have memory to store messages, but in the rewrite rules non-linear terms may occur, and thus, a principal can compare messages for equality; in the second extension, principals can store one message, but the rewrite rules may only contain linear terms. The first extension are the so-called rule-based protocols and the second extensions the linear-term one-memory protocols. In what follows, both extensions are defined and the undecidability results are proved.

4.1 Rule-based Protocols

A receive-send action will be defined by a set of input, output, and so-called process rules. A message is processed by such an action as follows: First one of the input rules is applied, resulting in a new message. Then, non-deterministically, process rules are applied to this message, and finally, one of the output rules is used to produce the actual output. In a more powerful model, one could allow a principal to produce output also when applying a process rule. However, to obtain a stronger undecidability result, we stick to the more restricted model.

Formally, as already mentioned, a *rewrite rule* is of the form $t \to t'$, where t and t' are terms. A *(rule-based) action* A of a principal is a tuple (I, O, R), where I, O, and

[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.

R are finite sets of rewrite rules (the input, output, and process rules, respectively). For every rule $t \to t' \in R$ we require that for all substitutions σ, $|\sigma(t')| < |\sigma(t)|$. This guarantees that process rules can only be applied to a message a finite number of times.

A rule-based action A defines the following binary relation R_A on $\mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon$: $(m, m') \in R_A$ iff there exist substitutions $\sigma_0, \ldots, \sigma_n$ and rewrite rules $r_0, \ldots, r_n$ with $r_i = t_i \to t'_i$, for every $i \leq n$, such that

- $r_0 \in I$, $r_n \in O$, and $r_i \in R$ for every $0 < i < n$;
- $\sigma_0(t_0) = m$; $\sigma_n(t'_n) = m'$; and
- $\sigma_i(t'_i) = \sigma_{i+1}(t_{i+1})$ for every $i < n$.

A generic protocol for which the receive-send actions can be described by rule-based actions is called *rule-based protocol.* Note that since in this setting principals do not have states, receive-send actions are simply subsets of $\mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon$ instead of $Q \times \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon \times Q$.

Theorem 4 *For rule-based protocols,* ATTACK *is undecidable.*

The proof shows that this theorem holds true even for protocols consisting only of one principal, which may perform only one receive-send action. In other words, the undecidability comes from the internal actions alone. Also, the problem is even undecidable if the intruder is not allowed to encrypt, decrypt, or hash messages.

The proof is rather straightforward. It is by reduction from *Post's Correspondence Problem* (PCP), which is defined as follows: Given an alphabet Σ with at least two letters and two sequences $u_0, \ldots, u_{n-1}$ and $v_0, \ldots, v_{n-1}$ of words over Σ (including the empty word ε), decide whether there exist indices $i_0, \ldots, i_{k-1}$, $k > 0$, such that $u_{i_0} \ldots u_{i_{k-1}} = v_{i_0} \ldots v_{i_{k-1}}$.

Given such a problem, we define the corresponding rule-based protocol P as follows: P has one principal performing one action $A = (I, O, R)$ with

- $I := \{x = x \to x = x\}$;
- $O := \{u_i = v_i \to \mathsf{secret} \mid i < k\}$;
- $R := \{u_i x = v_i y \to x = y \mid i < k\}$,

where x and y are variables. The initial intruder knowledge is $\mathcal{K} := \Sigma \cup \{=\}$. Now, it is easy to see that P allows a successful attack iff the instance of PCP has a solution. Obviously, the intruder does not need to encrypt, decrypt, or hash messages.

The reduction does not work if only linear terms are allowed in rewrite rules, because then one cannot compare submessages of arbitrary size for equality. More specifically, the rule $x = x \to x = x$ is not allowed, since $x = x$ is not a linear term. Nevertheless, if principals can store one message and this message can be compared with a submessage of the message being processed, we still have undecidability. This leads us to the linear-term one-memory protocols.

4.2 Linear-term One-memory Protocols

A *linear-term one-memory action* A is a tuple (I, O, R) such that

- I is a set of rules of the form $s \to (t, t')$, where s, t, t' are linear terms;
- O is a set of rules of the form $(s, s') \to t$, where s, s', t are linear terms;
- R is a set of rules of the form $(s, s') \to (t, t')$, where s, s', t, t' are linear terms and for all substitutions σ, $|\sigma(t)| + |\sigma(t')| < |\sigma(s)| + |\sigma(s')|$.

Note that the state of a principal only depends on the current input and is independent of previous receive-send actions. Therefore, similar to rule-based actions, a linear-term one-memory action $A = (I, O, R)$ induces a binary relation $R_A \subseteq \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon$ instead of a relation on $Q \times \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon \times Q$: $(m, m') \in R_A$ iff there exist substitutions $\sigma_0, \sigma_1, \ldots, \sigma_n$ and rules $r_0, \ldots, r_n$ such that

- $r_0 \in I$, $r_n \in O$, and $r_i \in R$ for every $0 < i < n$;
- $\sigma_0(s_0) = m$ if $r_0 = s_0 \to (t_0, t_0')$;
- $\sigma_n(t_n) = m'$ if $r_n = (s_n, s_n') \to t_n$; and
- $\sigma_i(t_i) = \sigma_{i+1}(s_{i+1})$ and $\sigma_i(t_i') = \sigma_{i+1}(s_{i+1}')$ if $r_i = (s_i, s_i') \to (t_i, t_i')$, for every $0 < i < n$, and $r_0 = s_0 \to (t_0, t_0')$.

A generic protocol for which the receive-send actions are define by linear-term one-memory actions is called *linear-term one-memory protocol*.

Theorem 5 *For linear-term one-memory protocols,* ATTACK *is undecidable.*

The proof is very similar to the one for rule-based protocols, and again is by a reduction from PCP. However, since we cannot test in one step whether the identity produced by the intruder in fact holds, this is done in two steps. First, one side of the identity is stored (input rule) and then in the next step, the stored term is compared with the other side of the identity (see rule $(x = y, y) \to (x = y, \#)$ below). Formally, given an instance of PCP as above, we define the corresponding linear-term one-memory protocol P as follows: P has one principal, who performs one action $A = (I, O, R)$ with

- $I := \{x = y \to (x = y, x)\}$;
- $O := \{(u_i = v_i, \#) \to \mathsf{secret} \mid i < k\}$;
- $R := \{(x = y, y) \to (x = y, \#)\} \cup \{(u_i x = v_i y, \#) \to (x = y, \#) \mid i < k\}$,

where x and y are variables, and $\# \notin \Sigma \cup \{=\}$ is a new letter. The initial intruder knowledge is $\mathcal{K} := \Sigma \cup \{=\}$. It is easy to see that P allows a successful attack iff the instance of PCP has a solution. Again, the intruder does not need to encrypt, decrypt, or hash messages.

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 for equality (which is possible if terms are not linear), or

2. store one message and compare this message 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. One such device is a transducer, i.e., a finite automaton with output.

In what follows, we define the instance of the generic protocol model, in which receive-send actions are modeled by transducers. 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; $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$.

Definition 6 *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 $q_0(v_0, w_0)q_1(v_1, w_1)q_2 \ldots (v_{n-1}, w_{n-1})q_n$ 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 $v_0 \ldots v_{n-1}$ is the *input label* and $w_0 \ldots w_{n-1}$ 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 lemma shows that it suffices to consider transducers with letter transitions.

Lemma 7 *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* $S, T \subseteq Q$*.*

PROOF. Every transition of $\mathcal{A}$ is turned into a set of transitions such that first the input label is read letter by letter and then the output label is written letter by letter. This requires to introduce intermediate states. Formally, let $(p, v, w, q) \in \Delta$ with $v = v_0 \cdots v_{l-1}$, $v_i \in \Sigma$ for all $i < l$, and $w = w_0 \cdots w_{r-1}$, $w_i \in \Omega$ for all $i < r$; $l = 0$ or $r = 0$ is allowed. Assume $l > 1$ or $r > 1$ (otherwise the transition has the correct format). This transition can be replaced by the transitions $(p_i, v_i, \varepsilon, p_{i+1})$, $i < l$, and $(q_i, \varepsilon, w_i, q_{i+1})$, $i < r$, where $p_0 = p$, $q_r = q$, $p_l = q_0$, and the states $p_1, \ldots, p_l, q_0, \ldots, q_{r-1}$ are new. Replacing every transition in this way yields a transducer with letter transitions, which satisfies the desired property. $\square$

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., "$)$". Often these letters will occur in expressions in the text (as in the definition of $\Sigma_{\mathcal{N}}$) without matching opening and closing parentheses, respectively. However, this should not lead to confusion.

A message transducer reads a message (interpreted as 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 8 *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). 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 Appendix B).

An open issue is whether these properties are decidable, i.e., given a transducer over $\Sigma_{\mathcal{N}}$ does it satisfy the properties. 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 2) 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 extended message transducers: $\mathcal{A} = (Q, \Sigma_{\mathcal{N}}, \Delta, (I_0, \ldots, I_n))$ 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 9 *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

We discuss 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 contains the main features. 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 Appendix B) 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 $\mathtt{hash}(m_0\mathtt{hash}(m_1 \cdots \mathtt{hash}(m_n)\cdots))$, where the m_i's are sequences of atomic messages and the nesting depth of the hashes is a priori unbounded (see Appendix A and B 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 message, and thus, well-formed.

Simulating rewrite rules. Transducers can simulate certain receive-send actions described by single rewrite rules. Consider for example the rule $\mathtt{enc}_k(x) \to \mathtt{hash}(kx)$, where x is a variable and k an atomic message: First, the transducer would read "$\mathtt{enc}_k($" and output "$\mathtt{hash}(k$", and then reads, letter by letter, the rest of the input message, i.e., "$x)$" – more precisely, the message substituted for x – and simultaneously writes it into the output. The transducer can also check whether the last letter of the input is a closing parenthesis; again, this is not necessary because all words sent to a message transducer (by the intruder) are well-formed.

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 other two restrictions. We will comment on this below.

First, we 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 (see Appendix B). 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 accepted by principals; this approach is usually pursued in protocol analysis based on finite-state model checking (e.g., [16]). For messages of bounded size all transformations performed by rewrite rules can also be carried out by message transducers.

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 Decidability Result

We show the following theorem.

Theorem 10 *For transducer-based protocols,* Attack *is decidable.*

To show decidability, we have to cope with two sources of infinite behavior. 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.

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 10, 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. To bound the number of internal actions, we apply a pumping argument showing that long paths in a message transducer can be truncated. This argument uses that principals (the extended message transducers describing them) have only finite memory.

More formally, we will show that the following problem is decidable. This immediately implies Theorem 10.

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

1. *$m_i \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{i-1}'\})$ for every $i < k$,*

2. $q_i^I(m_i, m_i')q_i^F \in \mathcal{A}_i$ *for every* $i < k$, *and*

3. $\mathsf{secret} \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{k-1}'\})$.

We write an instance of the PATHPROBLEM 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. The size of instances is defined as the size of the representation for $\mathcal{K}$ and $\mathcal{A}_0, \ldots, A_{k-1}$.

Using a pumping argument, we show that in order to find the messages m_i, m_i', for every $i < k$, it suffices to consider paths from q_i^I to q_i^F in $\mathcal{A}_i$ bounded in length by the size of the problem instance – the argument will also show that the bounds can be computed effectively. Thus, a decision procedure can enumerate all paths of length restricted by the (computed) bound and check whether their labels satisfy the conditions. (Note that for every message m and finite set $\mathcal{K}' \subseteq \mathcal{M}_\varepsilon$, $m \in \mathsf{d}(\mathcal{K}')$ can be decided.) In particular, as a "by-product" our decision procedure will yield an actual attack (if any).

The pumping argument: First, we define a *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 end, 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, it then remains 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.

In what follows, the argument is described in more detail. The formal definition of the orderings are, however, postponed to the subsequent sections in order to focus on the main ideas.

Preserving the solvability of instances of the path problem (cf. Section 6.1). For every $i \leq k$, we define a quasi-ordering[4] on messages $\preceq_i$ (the so-called solvability preserving ordering) which depends on the transducers $A_i, \ldots, A_{k-1}$ and has the following property (cf. Proposition 17): For every solvable instance $(\mathcal{K}, \mathcal{A}_i, \ldots, A_{k-1})$ of the path problem, every $m \in \mathcal{K}$, and $\overline{m} \in \mathcal{M}_\varepsilon$ with $m \preceq_i \overline{m}$, the instance $((\mathcal{K} \setminus \{m\}) \cup \{\overline{m}\}, \mathcal{A}_i, \ldots, A_{k-1})$ is solvable as well.

Assume that a path $q_i^I(m_i, m_i')q_i^F \in \mathcal{A}_i$ is replaced by a shorter path such that the corresponding input and output labels of the shorter path, say $\overline{m}_i$ and $\overline{m}_i'$, satisfy

[4] a reflexive and transitive ordering

$\overline{m}_i \in \mathsf{d}(\mathcal{K} \cup \{m'_0, \ldots, m'_{i-1}\})$ and $m'_i \preceq_{i+1} \overline{m}'_i$. Then, after $\mathcal{A}_i$ has returned $\overline{m}'_i$ on input $\overline{m}_i$, the resulting intruder knowledge is $\mathcal{K} \cup \{m'_0, \ldots, m'_{i-1}, \overline{m}'_i\}$ instead of $\mathcal{K} \cup \{m'_0, \ldots, m'_{i-1}, m'_i\}$. Using Proposition 17, we conclude that there still exists a solution for the rest of the instance, i.e., $(\mathcal{K} \cup \{m'_0, \ldots, m'_{i-1}, \overline{m}'_i\}, \mathcal{A}_{i+1}, \ldots, A_{k-1})$.

Consequently, it remains to find criteria for truncating long paths such that $\overline{m}_i \in \mathsf{d}(\mathcal{K} \cup \{m'_0, \ldots, m'_{i-1}\})$ and $m'_i \preceq_{i+1} \overline{m}'_i$. Truncating paths such that the condition on $\overline{m}_i$ is satisfied is rather easy. The involved part is the condition on the output label $\overline{m}'_i$. To this end, we introduce the path truncation ordering. We also need to show that the depth of messages in solutions of the problem instance can be bounded.

Truncating paths (cf. Section 6.4). We extend $\preceq_i$ to a quasi-ordering $\preceq^l_i$ (the path truncation ordering) on so-called left half-messages. Left half-messages are prefixes of messages (considered as words over $\Sigma_{\mathcal{N}}$). In particular, left half-messages may lack some closing parentheses. The "l" in $\preceq^l_i$ is the number of missing parentheses (the *level* of left half-messages); $\preceq^l_i$ only relates left half-messages of level l. Analogously, right half-messages are suffixes of messages. Thus, they may have too many closing parentheses; the number of additional parentheses determines the level of right half-messages. The equivalence relation $\equiv^l_i$ on left half-messages corresponding to $\preceq^l_i$ has the following property (Proposition 29): For all left half-messages α, α' of level l and right half-messages γ of level l, $\alpha \equiv^l_i \alpha'$ implies $\alpha\gamma \equiv_i \alpha'\gamma$. (Note that $\alpha\gamma$ and $\alpha'\gamma$ are messages.)

Now, consider two positions $x < y$ in the path $\pi = (q^I_i, m_i, m'_i, q^F_i) \in \mathcal{A}_i$ such that α_x, α_y are the output labels up to these positions, and γ_x, γ_y are the output labels beginning at these positions, i.e., $m'_i = \alpha_x\gamma_x = \alpha_y\gamma_y$. Clearly, α_x, α_y are left half-messages and γ_x, γ_y are right half-messages. Assume that α_x, α_y have the same level l (in particular, γ_x, γ_y have level l) and $\alpha_x \equiv^l_i \alpha_y$. Then, by Proposition 29, it follows $m'_i = \alpha_y\gamma_y \equiv_i \alpha_x\gamma_y$, where $\alpha_x\gamma_y$ is the output label of the path obtained by cutting out the subpath in π between x and y.[5] Thus, $\equiv^l_i$ provides us with the desired criterion for "safely" (in the sense of 1.) truncating paths. In order to conclude that the length of paths can be bounded in the size of the problem instance, it remains to show that l and the index of $\equiv^l_i$ (i.e., the number of equivalence classes modulo $\equiv^l_i$ on left half-messages of level l) can be bounded in the size of the problem instance. To this end, the following is shown.

Bounding the depth of messages (cf. Section 6.2 and 6.3). We first show that if $(m_0, m'_0, \ldots, m_{k-1}, m'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$, then, for every i, their also exists a solution if the depth of m_i is bounded in the size of the problem instance (Proposition 20).

We then show how the depth of the output message m'_i can be bounded: Let π be a path in $\mathcal{A}_i$ from q^I_i to q^F_i or a strict path in $\mathcal{A}_i$, and x be a position in π such that α_x is the input label of π up to position x and β_x is the output label of π up to

[5] One little technical problem is that $\alpha_x\gamma_y$ does not need to be a message since it may contain a word of the form $\mathtt{enc}_a()$, which is not a message. However, if one considers three positions $x < y < z$, then one can show that either $\alpha_x\gamma_y$ or $\alpha_y\gamma_z$ is a message.

x. Then, the level of β_x can be bounded by a polynomial in the level of α_x and the number of states of $\mathcal{A}_i$ (Proposition 21). As a corollary, one obtains that the depth of output messages can be bounded in the depth of input messages (Corollary 22), and using Proposition 20, that both the depth of input and output messages can be bounded in the size of the problem instance (Corollary 23).

With this, one can show that the index of $\equiv_i^l$ is bounded (Proposition 31). Moreover, because of Corollary 23, the l in 2. (the level of the half-messages α_x, α_y, γ_x, γ_y) is bounded in the size of the problem instance. Therefore, $\equiv_i^l$ can serve as the desired criterion for truncating paths.

Following these steps, we now provide a detailed proof. We will define the different orderings and prove the needed properties. In Section 6.5 everything will be put together to show decidability of the path problem.

In order to simplify the presentation, we will not consider hashing, i.e., from now on, $\Sigma_{\mathcal{N}}$ does not contain the symbol "`hash(`". However, all definitions and results easily carry over to the more general case.

6.1 The Solvability Preserving Ordering $\preceq_i$

In this section, we formally define the solvability preserving ordering $\preceq_i$. In Section 6.1.2 we show that $\preceq_i$ is in fact solvability preserving. To do so, we first need to prove that $\preceq_i$ is closed under substitution (Section 6.1.1). Finally, it is shown that the index of the equivalence relation induced by $\preceq_i$ is finite (Section 6.1.3).

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 $\mathsf{an}(\{m_0, \ldots, m_{n-1}\} \cup \mathcal{K}) \cap \mathcal{K}'$. For the definition of $\preceq_i$, we need the ordering $\preceq$.

Definition 11 *For messages $m, m' \in \mathcal{M}_\varepsilon$, we define $m \preceq m'$ 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 $a, a', a'' \in \mathcal{N}$, $\mathtt{enc}_a(\mathtt{enc}_{a'}(a'')) \preceq a'\mathtt{enc}_{a'}(\mathtt{enc}_a(a''))$. Obviously, $\preceq$ is a quasi-ordering, i.e., it is reflexive and transitive.

The relation $\preceq_i$, $i \leq k$, depends on the message transducers $\mathcal{A}_i, \ldots, \mathcal{A}_{k-1}$ and is defined inductively. The definition also needs the ordering $\sqsubseteq_i$. To understand the definition of $\preceq_i$ recall that we want to guarantee that if $(\mathcal{K}, \mathcal{A}_i, \ldots, A_{k-1})$ has a solution, say $(m_i, m_i', \ldots, m_{k-1}, m_{k-1}')$, then so has $(\mathcal{K} \setminus \{m\} \cup \{m'\}, \mathcal{A}_i, \ldots, A_{k-1})$. 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 2. in the definition below. We also need that $\preceq_i$ refines $\preceq_{i+1}$ (1.). Finally, in our proofs we will use that $\preceq_i$ is closed under substitution. With condition 3., this can be shown.

Definition 12 *For every $i \leq k$, the* solvability preserving ordering *$\preceq_i$ is defined as follows: for all $m, m' \in \mathcal{M}_\varepsilon$, $m \preceq_i m'$ iff i) $m = m' = \varepsilon$, or ii) $m \neq \varepsilon$, $m' \neq \varepsilon$, $m \preceq m'$, and if $i < k$, then*

1. $m \preceq_{i+1} m'$;

2. *for every $N \subseteq \mathcal{N}$ and $x \in \mathsf{an}(m, N)$ with $x = \mathbf{enc}_a(z)$ for some $a \in \mathcal{N}$ and $z \in \mathcal{M}$, there exists $x' := \mathbf{enc}_{a'}(z') \in \mathsf{an}(m', N)$ for some $a' \in \mathcal{N}$ and $z' \in \mathcal{M}$ such that $x \sqsubseteq_i x'$; and*

3. $m \sqsubseteq_i m'$.

For $i < k$ and messages $m, m' \in \mathcal{M}_\varepsilon$, we define $m \sqsubseteq_i m'$ iff i) $m = m' = \varepsilon$, or ii) $m \neq \varepsilon$, $m' \neq \varepsilon$, and for every $p, q \in Q_i$ and $y \in \mathcal{M}_\varepsilon$, $p(m, y)q \in_s \mathcal{A}_i$ implies that there exists $y' \in \mathcal{M}_\varepsilon$ with $p(m', y')q \in_s \mathcal{A}_i$ and $y \preceq_{i+1} y'$.

The following lemma is proved by a simple induction on $i \leq k$.

Lemma 13 *For every $i \leq k$, $\preceq_i$ is a quasi-ordering.*

6.1.1 Closure under Substitution

To show that $\preceq_i$ in fact preserves solvability (in the sense explained above), we first show that $\preceq_i$ is closed under substitution. This is done by induction on $i \leq k$. The base case, $i = k$, amounts to showing that $\preceq$ is closed under substitution. This requires some notation.

For a set $V = \{v_0, \ldots, v_{n-1}\}$ of variables and a subset $T \subseteq \mathcal{T}_\varepsilon(V)$ of terms over V, we define $\mathsf{an}_c(T)$ to be the closure of T under decomposition, $\mathsf{an}_e(T)$ to be the closure of T under decryption, and $\mathsf{an}_{cec}(T)$ the closure of T under decomposition, encryption, and decomposition (in this order), where "c" stands for composition and "e" for encryption. Formally,

$$
\begin{aligned}
\mathsf{an}_c(T) &:= \{y \in \mathcal{M} \mid \text{ there exist } x, z \in \mathcal{M}_\varepsilon \text{ with } xyz \in T\},\\
\mathsf{an}_e(T) &:= \{x \in \mathcal{M} \mid \text{ there exist } a \in T \cap \mathcal{N} \text{ with } \mathsf{enc}_a(x) \in T\} \cup T\\
\mathsf{an}_{cec}(T) &:= \mathsf{an}_c(\mathsf{an}_e(\mathsf{an}_c(T))).
\end{aligned}
$$

It is easy to see that

$$
\mathsf{an}(T) \quad = \quad \bigcup_{i \geq 0} \mathsf{an}^i_{cec}(T), \tag{1}
$$

where $\mathsf{an}^0_{cec}(T) := T$ and $\mathsf{an}^{i+1}_{cec}(T) := \mathsf{an}_{cec}(\mathsf{an}^i_{cec}(T))$.

We abbreviate $\mathsf{an}^i_{cec}(\{t_0, \ldots, t_{n-1}\} \cup T)$ by $\mathsf{an}^i_{cec}(t_0, \ldots, t_{n-1}, T)$, and $\mathsf{an}^i_{cec}(t_0, \ldots, t_{n-1}, T) \cap \mathcal{N}$ by $\mathsf{an}^i_{cec,\mathcal{N}}(t_0, \ldots, t_{n-1}, T)$.

Given a term t and $N \subseteq \mathcal{N}$, we say that a subterm t' of t is *N-accessible*[6] if

1. $t' \in \mathsf{an}_c(t)$; or

2. there exists $\mathsf{enc}_a(t'') \in \mathsf{an}_c(t)$, $a \in N$, and t' is N-accessible in t''.

Lemma 14 *Let $x_0, \ldots, x_{n-1}, x'_0, \ldots, x'_{n-1} \in \mathcal{M}$ be messages and $t(v_0, \ldots, v_{n-1})$ be a term. If $x_i \preceq x'_i$ for all $i < n$, then $t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}] \preceq t[v_0/x'_0, \ldots, v_{n-1}/x'_{n-1}]$.*

[6]This notion was already defined in [2]

PROOF. Define $m := t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$ and $m' := t[v_0/x'_0, \ldots, v_{n-1}/x'_{n-1}]$. Because of (1) it suffices to show

$$\mathsf{an}^i_{cec,\mathcal{N}}(m, N) \subseteq \mathsf{an}_{\mathcal{N}}(m', N)$$

for every $i \geq 0$ and $N \subseteq \mathcal{N}$.

Assume $i = 0$ and $a \in (\{m\} \cup N) \cap \mathcal{N}$. If $a \in N$, nothing is to show. Otherwise, $t = a$ or there exists $i < n$ with $x_i = a$ (i.e., $t = v_i$). In the former case it immediately follows that $a \in \mathsf{an}_{\mathcal{N}}(m', N)$. In the latter case, $x_i \preceq x'_i$ implies $x'_i = a$, and thus, $a \in \mathsf{an}_{\mathcal{N}}(m', N)$.

Assume $i > 0$ and $a \in \mathsf{an}_{cec}(\mathsf{an}^i_{cec}(m, N)) \cap \mathcal{N}$. If $a \in \mathsf{an}^i_{cec}(m, N)$, the induction hypothesis yields $a \in \mathsf{an}_{\mathcal{N}}(m', N)$. Otherwise, there must exist $x \in \mathsf{an}^i_{cec}(m, N)$, a message z, and $b \in \mathsf{an}^i_{cec}(m, N) \cap \mathcal{N}$ with $x = \mathsf{enc}_b(z)$ and $a \in \mathsf{an}_c(z)$. We distinguish two cases.

i) There exists a term t' such that $\mathsf{enc}_b(t')$ is a subterm of t, $z = t'[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$, and t' is $(\mathsf{an}^i_{cec,\mathcal{N}}(m, N))$-accessible in t. Thus, by the induction hypothesis, t' is $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in t. Consequently, if $a \in \mathsf{an}_c(t')$, then $a \in \mathsf{an}_{\mathcal{N}}(m', N)$. Otherwise, there must exist $i < n$, $t_0, t_1 \in \mathcal{T}_\varepsilon(V)$ such that $t' = t_0 v_i t_1$, and $a \in \mathsf{an}_c(\{x_i\})$. Because $x_i \preceq x'_i$, it follows $a \in \mathsf{an}(x'_i)$, and therefore, $a \in \mathsf{an}_{\mathcal{N}}(m', N)$.

ii) There exists $i < n$ such that x, and thus z, is a submessage of x_i. Let $N' := \mathsf{an}^i_{cec,\mathcal{N}}(m, N)$. By induction hypothesis $N' \subseteq \mathsf{an}_{\mathcal{N}}(m', N)$. We know that z and x_i are N'-accessible in m. Thus, x'_i is N'-accessible in m', and therefore also $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible. Now $x_i \preceq x'_i$ implies $\mathsf{an}(x_i, N') \subseteq \mathsf{an}(x'_i, N')$, and we know $a \in \mathsf{an}(x_i, N')$. Thus, $a \in \mathsf{an}(x'_i, N')$. With $N' \subseteq \mathsf{an}_{\mathcal{N}}(m', N)$ this yields $a \in \mathsf{an}(x'_i, \mathsf{an}_{\mathcal{N}}(m', N))$. Finally, since x'_i is $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in m', we obtain $a \in \mathsf{an}_{\mathcal{N}}(m', N)$. □

We generalize Lemma 14 to the solvability preserving ordering $\preceq_i$.

Lemma 15 *Let $x_0, \ldots, x_{n-1}, x'_0, \ldots, x'_{n-1} \in \mathcal{M}_\varepsilon$, $i \leq k$, and $t(v_0, \ldots, v_{n-1})$ be a term, where every variable v_i occurs at most once in t. If $x_j \preceq_i x'_j$ for all $j < n$, then $t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}] \preceq_i t[v_0/x'_0, \ldots, v_{n-1}/x'_{n-1}]$.*

PROOF. W.l.o.g., we can assume that $x_j \neq \varepsilon$ and $x'_j \neq \varepsilon$ for all $j < n$, since otherwise $x_j = x'_j = \varepsilon$, and we can remove v_j from t altogether.

The proof is by induction on i. If $k = i$, the statement follows immediately from Lemma 14. Define $m := t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$ and $m' := t[v_0/x'_0, \ldots, v_{n-1}/x'_{n-1}]$. Assume $i < k$. Following Definition 12, we show $m \preceq_i m'$.

1. From $x_j \preceq_i x'_j$, $j < n$, it follows $x_j \preceq_{i+1} x'_j$. Thus, by the induction hypothesis, $m \preceq_{i+1} m'$.

2. Let $N \subseteq \mathcal{N}$ and $x \in \mathsf{an}(m, N)$ with $x = \mathsf{enc}_a(z)$ for some message z and $a \in \mathcal{N}$. Note that since x is of the form $\mathsf{enc}_a(\cdot)$, it cannot happen that just part of an x_i belongs to x, and therefore, it suffices to consider the two following cases.

   i) There exists a subterm t' of t such that $x = t'[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$ and $t' \notin \{v_0, \ldots, v_{n-1}\}$. Let $x' := t'[v_0/x'_0, \ldots, v_{n-1}/x'_{n-1}]$. We know that t' is $\mathsf{an}_{\mathcal{N}}(m, N)$-accessible in t. Since $m \preceq m'$, t' is also $\mathsf{an}_{\mathcal{N}}(m', N)$-accessible in t.

In particular, $x' \in \mathsf{an}(m', N)$. Since t' is not a variable, it follows that t' is of the form $\mathtt{enc}_a(t'')$ for some term t'' over $\{v_0, \ldots, v_{n-1}\}$. Thus, x' has the form $\mathtt{enc}_a(z')$ for some message z'. It remains to show that $x \sqsubseteq_i x'$.

Let $p, q \in Q_i$, $y \in \mathcal{M}_\varepsilon$ with $\pi := p(x, y)q \in_s A_i$. We know that x is of the form $t'[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$. Thus, if an v_j occurs in t', then π contains a subpath of the form $p_j(x_j, y_j)p'_j \in_s \mathcal{A}_i$. The definition of message transducer guarantees that $y_j \in \mathcal{M}_\varepsilon$. Moreover, there exists a term $t''(v_0, \ldots, v_{n-1})$, where every variable occurs at most once, such that $y = t''[v_0/y_0, \ldots, v_{n-1}/y_{n-1}]$.

Because $x_j \preceq_i x'_j$, there exists $y'_j \in \mathcal{M}_\varepsilon$ with $p_j(x'_j, y'_j), p'_j \in_s \mathcal{A}_i$ and $y_j \preceq_{i+1} y'_j$. Set $y' := t''[v_0/y'_0, \ldots, v_{n-1}/y'_{n-1}]$. By induction hypothesis, $y \preceq_{i+1} y'$. Furthermore, replacing in π the subpaths $p_j(x_j, y_j)p'_j$ by $p_j(x'_j, y'_j), p'_j$ shows that $p(x', y')q \in_s A_i$.

ii) There exists $j < n$ such that x is a subterm of x_j. In particular, x_j is $\mathsf{an}_\mathcal{N}(m, N)$-accessible in m. Thus, because $\mathsf{an}_\mathcal{N}(m, N) \subseteq \mathsf{an}_\mathcal{N}(m', N)$, x'_j is $\mathsf{an}_\mathcal{N}(m', N)$-accessible in m'. We also know that $x \in \mathsf{an}(x_j, \mathsf{an}_\mathcal{N}(m, N))$. Then, $x_j \preceq_i x'_j$ implies that there exists $x' \in \mathsf{an}(x'_j, \mathsf{an}_\mathcal{N}(m, N))$ of the form $\mathtt{enc}_b(z')$ for some message z' and $b \in \mathcal{N}$ with $x \sqsubseteq_i x'$. In particular, $x' \in \mathsf{an}(x'_j, \mathsf{an}_\mathcal{N}(m', N))$, and given that x'_j is $\mathsf{an}_\mathcal{N}(m', N)$-accessible in m', it follows $x' \in \mathsf{an}(m', N)$.

3. Similar to 2.,i), one shows $m \sqsubseteq_i m'$. □

6.1.2 The Ordering $\preceq_i$ is Solvability Preserving

We show that $\preceq_i$ is solvability preserving by induction on $i \leq k$. The base case, $i = k$, is a consequence of the following lemma.

Lemma 16 *For all $m, m' \in \mathcal{M}$, $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$, if $m \preceq m'$, then $\mathsf{an}_\mathcal{N}(m, \mathcal{K}) \subseteq \mathsf{an}_\mathcal{N}(m', \mathcal{K})$.*

PROOF. The proof is very similar to the one for Lemma 14. We show $\mathsf{an}^i_{cec,\mathcal{N}}(m, \mathcal{K}) \subseteq \mathsf{an}_\mathcal{N}(m', \mathcal{K})$ for every $i \geq 0$ by induction on i.

Assume $i = 0$ and $a \in (\{m\} \cup \mathcal{K}) \cap \mathcal{N}$. If $a \in \mathcal{K}$, nothing is to show. Otherwise, $m = a$, and $m \preceq m'$ implies $\mathsf{an}_\mathcal{N}(m) \subseteq \mathsf{an}_\mathcal{N}(m')$, and thus, $a \in \mathsf{an}_\mathcal{N}(m', \mathcal{K})$.

Assume $i > 0$ and $a \in \mathsf{an}_{cec}(\mathsf{an}^i_{cec}(m, \mathcal{K})) \cap \mathcal{N}$. If $a \in \mathsf{an}^i_{cec}(m, \mathcal{K})$, the induction hypothesis yields $a \in \mathsf{an}_\mathcal{N}(m', \mathcal{K})$. Otherwise, there must exist $x \in \mathsf{an}^i_{cec}(m, \mathcal{K})$, a message z, and $b \in \mathsf{an}^i_{cec,\mathcal{N}}(m, \mathcal{K})$ with $x = \mathtt{enc}_b(z)$ and $a \in \mathsf{an}_c(z)$. We distinguish two cases.

i) The messages x and z are submessage of some message x' in $\mathcal{K}$. In particular, x and z are $\mathsf{an}^i_{cec,\mathcal{N}}(m, \mathcal{K})$-accessible in x', and thus, x and z are $\mathsf{an}_\mathcal{N}(m', \mathcal{K})$-accessible in x'. Consequently, $a \in \mathsf{an}_\mathcal{N}(m', \mathcal{K})$.

ii) The messages x and z are submessages of m. Let $N := \mathsf{an}^i_{cec,\mathcal{N}}(m, \mathcal{K})$. By induction hypothesis $N \subseteq \mathsf{an}_\mathcal{N}(m', \mathcal{K})$. We know $a \in \mathsf{an}_\mathcal{N}(m, N)$, and $m \preceq m'$ implies $\mathsf{an}_\mathcal{N}(m, N) \subseteq \mathsf{an}_\mathcal{N}(m', N)$. Thus, $a \in \mathsf{an}_\mathcal{N}(m', N) \subseteq \mathsf{an}_\mathcal{N}(m', \mathcal{K})$. □

Using Lemma 16 and 15, we prove the main statement of this subsection.

Proposition 17 *Assume that* $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$, $i \leq k$, *is a solvable instance of* PATHPROBLEM *and* $m \in \mathcal{K}$. *Then, for every* $\overline{m} \in \mathcal{M}_\varepsilon$ *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.*

PROOF. The proof is by induction on $i \leq k$. If $m = \varepsilon$, then, by definition of $\preceq_i$, $\overline{m} = \varepsilon$ and nothing has to be shown. Therefore, we assume that $m \neq \varepsilon$, and thus, $\overline{m} \neq \varepsilon$. The induction basis, $i = k$, immediately follows from Lemma 16.

Now assume $i < k$. Define $N := \mathsf{an}_\mathcal{N}(\mathcal{K})$ and $M := \{\mathsf{enc}_a(z) \in \mathsf{an}(m, N) \mid$ there exist $z \in \mathcal{M}$ and $a \in \mathcal{N}\}$. 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, $m_i \in \mathsf{d}(\mathcal{K})$. If in an derivation for m_i, a message of the form $x\mathsf{enc}_a(z)y \in \mathsf{an}(m, N)$ is used to construct m_i, we will w.l.o.g. assume that the single messages $x, \mathsf{enc}_a(z), y$ are used, since all three messages belong to $\mathsf{an}(m, N)$. Let n be the number of times a message in M was used to derive m_i from $\mathcal{K}$, and let $\{x_0, \ldots, x_{n-1}\}$ be the multiset of these messages; an $x_j \in M$ occurs in this multiset as many times as it was used in the derivation of m_i. (Due to the assumption on derivations made before, every occurrence of some message in M is taken into account.) Also, let $v_0, \ldots, v_{n-1}$ be pairwise distinct variables. Then, there exists a term t over $\{v_0, \ldots, v_{n-1}\}$ such that $m_i = t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$. Since $m \preceq_i \overline{m}$, for every x_j, $j < n$, there exists $\overline{x}_j \in \mathsf{an}(\overline{m}, N)$ with $x_j \sqsubseteq_i \overline{x}_j$. Define $\overline{m}_i := t[v_0/\overline{x}_0, \ldots, v_{n-1}/\overline{x}_{n-1}]$. Since, by Lemma 16, $N \subseteq \mathsf{an}(\overline{\mathcal{K}})$, we can conclude $\overline{x}_j \in \mathsf{an}(\overline{\mathcal{K}})$, and it follows $\overline{m}_i \in \mathsf{d}(\overline{\mathcal{K}})$, since the derivation of $\overline{m}_i$ from $\overline{\mathcal{K}}$ coincides with the one for m_i from $\mathcal{K}$ except that the x_js are replaced by $\overline{x}_j$.

In $\pi = q_i^I(m_i, m_i')q_i^F \in \mathcal{A}_i$, there exist subpaths of the form $p_j(x_j, y_j)p_{j+1} \in_s \mathcal{A}_i$, for every $j < n$. By the properties of message transducers, we know that $y_j \in \mathcal{M}_\varepsilon$, and there exists a term t' over $\{v_0, \ldots, v_{n-1}\}$, where every v_j, $j < n$, occurs exactly once in t', with $m_i' = t'[v_0/y_0, \ldots, v_{n-1}/y_{n-1}]$. Since $x_j \sqsubseteq_i \overline{x}_j$, there exists $\overline{y}_j$ with $p_j(\overline{x}_j, \overline{y}_j)p_{j+1} \in_s \mathcal{A}_i$ and $y_j \preceq_{i+1} \overline{y}_j$. Define $\overline{m}_i' := t'[v_0/\overline{y}_0, \ldots, v_{n-1}/\overline{y}_{n-1}]$.

If we replace in π every subpath $p_j(x_j, y_j)p_{j+1} \in_s \mathcal{A}_i$ by $p_j(\overline{x}_j, \overline{y}_j)p_{j+1} \in_s \mathcal{A}_i$, we obtain $q_i^I(\overline{m}_i, \overline{m}_i')q_i^F \in \mathcal{A}_i$.

By Lemma 15, we have $m_i' \preceq_{i+1} \overline{m}_i'$. Thus, since $(\mathcal{K} \cup \{m_i'\}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$ has a solution, by induction hypothesis, the instance $(\mathcal{K} \cup \{\overline{m}_i'\}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$ is also solvable. Finally, since $m \preceq_i \overline{m}$ implies $m \preceq_{i+1} \overline{m}$, the induction hypothesis also yields a solution for $(\overline{\mathcal{K}} \cup \{\overline{m}_i'\}, \mathcal{A}_{i+1}, \ldots, A_{k-1})$, and together with $\overline{m}_i$ and $\overline{m}_i'$, this is a solution for $(\overline{\mathcal{K}}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$. □

6.1.3 The Index of the Solvability Preserving Equivalence Relation

If $\leq$ is a quasi-ordering on a set S, then the relation $\equiv$ with $a \equiv b$ iff $a \leq b$ and $b \leq a$ for all $a, b \in S$ is an equivalence relation on S. The *equivalence class* of a modulo $\equiv$ is denoted $[a]_\equiv := \{b \mid a \equiv b\}$. The *index* $I(\equiv)$ of $\equiv$ is the number of different equivalence classes over S.

In what follows, let $\equiv$, $\equiv_i$, $=_i$ denote the equivalence relations corresponding to $\preceq$, $\preceq_i$, and $\sqsubseteq_i$.

We show that the index of $\equiv_i$ is finite. To this end, we first consider $\equiv$. The proof of the following lemma is straightforward.

Lemma 18 *The index of $\equiv$ is finite. More precisely,*

$$I(\equiv) \in \mathcal{O}(2^{2^{|\mathcal{N}|}\cdot|\mathcal{N}|})$$

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

Proposition 19 *The index of $\equiv_k$, and for every $i < k$, the index of $\equiv_i$ and $=_i$ is finite, and can be bounded as follows:*

$$\begin{array}{rcl} I(=_i) & \in & \mathcal{O}(2^{I(\equiv_{i+1})\cdot 2^{|Q_i|^2}}) \\ I(\equiv_i) & \in & \mathcal{O}(I(\equiv_{i+1})\cdot 2^{I(=_i)\cdot 2^{|\mathcal{N}|}}\cdot I(=_i)) \end{array}$$

Proof. The proof is by induction on i. For $i = k$, Lemma 18 shows that the index of $\equiv_k$ is finite. Assume that $i < k$ and the index of $\equiv_{i+1}$ is finite. We first show that the index of $=_i$ is finite and from this conclude that $\equiv_i$ has finite index.

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

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

To prove that $=_i^t$ has finite index, consider the mapping f_i^t which takes every tuple (x, y) to the tuple $([y]_{\equiv_{i+1}}, \{(p, q) \mid p(x, y)q \in_s \mathcal{A}_i\})$. Since $\equiv_{i+1}$ has a finite index, it follows that the range of f_i^t is finite. Moreover, it is easy to see that $f_i^t(x, y) = f^t(x', y')$ implies $(x, y) =_i^t (x', y')$. From this, it immediately follows that $=_i^t$ has finite index, and that

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

To show that $=_i$ has finite index, define $M_{i,x} := \{[(x, y)]_{=_i^t} \mid y \in \mathcal{M}_\varepsilon\}$. Clearly, $M_{i,x}$ is a finite set, and there are only finitely many different sets $M_{i,x}$. Together with the following claim, this implies that $=_i$ has finite index.

Claim I. For messages $x, x' \in \mathcal{M}$, $M_{i,x} = M_{i,x'}$ implies $x =_i x'$.

Proof of the claim. Assume $M_{i,x} = M_{i,x'}$. We show $x \sqsubseteq_i x'$; $x' \sqsubseteq_i x$ follows by symmetry. Let $p, q \in Q_i$ and $y \in \mathcal{M}_\varepsilon$ with $p(x, y)q \in_s \mathcal{A}_i$. We know $[(x, y)]_{=_i^t} \in M_{i,x} = M_{i,x'}$. Thus, there exists $y' \in \mathcal{M}_\varepsilon$ such that $[(x, y)]_{=_i^t} = [(x', y')]_{=_i^t}$. Consequently, by definition of $=_i^t$, $y \equiv_{i+1} y'$ and $p(x', y')q \in_s \mathcal{A}_i$. This shows $x \sqsubseteq_i x'$, and concludes the proof of the claim.

From this it immediately follows that

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

We now show that the index of $\equiv_i$ is finite. Let $M_{i,m,N} := \{[x]_{=_i} \mid x \in \mathsf{an}(m, N)$ and $x = \mathtt{enc}_a(z)$ for $z \in \mathcal{M}$ and $a \in \mathcal{N}\}$, and f_i be a mapping that takes every message $m \in \mathcal{M}$ to $([m]_{\equiv_{i+1}}, (M_{i,m,N} \mid N \subseteq \mathcal{N}), [m]_{=_i})$. We know that the index of $\equiv_{i+1}$

and $=_i$ is finite. Thus, $M_{i,m,N}$ is finite and there are only finitely many different sets $M_{i,m,N}$. Also, recall that $\mathcal{N}$ is a finite set. As a consequence, the range of f_i is finite. Together with the following claim, we can conclude that $\equiv_i$ has finite index.

Claim II. For messages $m, m' \in \mathcal{M}$, $f_i(m) = f_i(m')$ implies $m \equiv_i m'$.

Proof of the claim. Assume $f_i(m) = f_i(m')$. We show $m \preceq_i m'$; $m' \preceq_i m$ follows by symmetry.

1. From $f_i(m) = f_i(m')$, we immediately obtain $m \preceq_{i+1} m'$, and in particular, $m \preceq m'$.
2. Let $N \subseteq \mathcal{N}$, $x \in \mathsf{an}(m, N)$ with $x = \mathsf{enc}_a(z)$ for some $z \in \mathcal{M}$ and $a \in \mathcal{N}$. Thus, $[x]_{=_i} \in M_{i,m,N} = M_{i,m',N}$. Consequently, there exists $x' \in \mathsf{an}(m', N)$ with $x' = \mathsf{enc}_b(z')$ for some $z' \in \mathcal{M}$, $b \in \mathcal{N}$, and $[x']_{=_i} = [x]_{=_i}$. In particular, $x \sqsubseteq_i x'$.
3. From $f_i(m) = f_i(m')$, $m =_i m'$ follows immediately, and thus, $m \sqsubseteq_i m'$.

This concludes the proof of the claim.

As an immediate consequence, we obtain that

$$I(\equiv_i) \in \mathcal{O}(I(\equiv_{i+1}) \cdot 2^{I(=_i) \cdot 2^{|\mathcal{N}|}} \cdot I(=_i))$$

□

6.2 Bounding the Depth of Input Messages

We show the following proposition.

Proposition 20 *Assume that $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$ is a solution of the instance $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ of the path problem. Then, there exists a solution $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ of this instance with* $\mathsf{depth}(\overline{m}_i) \leq 2 \cdot |Q_i|^2 \cdot I(\equiv_{i+1}) + \mathsf{depth}(\mathcal{K}) + 2$.

In the following subsection, we will show that the depth of m'_i can be bounded as well. As an immediate consequence, we will obtain that to find solutions for the path problem it suffices to only consider messages of depth bounded in the size of the problem instance.

To prove Proposition 20, we need some notation. A word $\alpha \in \Sigma^*_{\mathcal{N}}$ is a *left half-message*, if α is a prefix of a message, i.e., there exists a word $\gamma \in \Sigma^*_{\mathcal{N}}$ such that $\alpha\gamma$ is a message. In Section 6.4, we also consider *right half-messages*, i.e., suffices of messages. For a left half-message α, the *level $l(\alpha)$ of α* is defined as the number of symbols "$\mathsf{enc}_a($", for some $a \in \mathcal{N}$, without matching closing parentheses. Analogously, for a right half-message γ, the *level $l(\gamma)$ of γ* is the number of closing parenthesis in γ without a matching encryption symbol "$\mathsf{enc}_a($", for some $a \in \mathcal{N}$.

Now, let $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$ be a solution of the instance $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ of the path problem. Let π denote a path

$$q_0(a_0, b_0)q_1(a_1, b_1) \cdots (a_{r-1}, b_{r-1})q_r \tag{2}$$

in $\mathcal{A}_i$ with $r > 0$, $a_j, b_j \in \Sigma_{\mathcal{N}} \cup \{\varepsilon\}$, for every $j < r$, $q_0 = q_i^I$, $q_r = q_i^F$, $m_i = a_0 \cdots a_{r-1} \in \mathcal{M}_\varepsilon$, and $m'_i = b_0 \cdots b_{r-1} \in \mathcal{M}_\varepsilon$. We define $l_\pi(i) := l(a_0 \cdots a_{i-1})$ and $l'_\pi(i) := l(b_0 \cdots b_{i-1})$ for all $i \leq r$, to be the *input and output level function of* π, respectively. Let $m_i(j, j') := a_j \cdots a_{j'-1}$ and $m'_i(j, j') := b_j \cdots b_{j'-1}$ for all $0 \leq j \leq j' \leq r$.

Assume $\mathsf{depth}(m_i) > 2 \cdot |Q_i|^2 \cdot I(\equiv_{i+1}) + \mathsf{depth}(\mathcal{K}) + 2$. Then, there exist three nested encryptions $\mathtt{enc}_{c_u}(z_u)$ in m_i for some $c_u \in \mathcal{N}$ and $z_u \in \mathcal{M}$, $u \in \{0, 1, 2\}$, with the following properties. There exist positions $i_0, j_0, i_1, j_1, i_2, j_2$ in π with $i_0 < i_0 < i_1 < i_2 < j_2 < j_1 < j_0 \leq r$ such that

- $a_{i_u} =$ "$\mathtt{enc}_{c_u}($" and $a_{j_u - 1} =$ "$)$" for some $c_u \in \mathcal{N}$ and $u \in \{0, 1, 2\}$; in particular $m_i(i_u, j_u)$ is of the form $\mathtt{enc}_{c_u}(z_u)$ for some $z_u \in \mathcal{M}$;
- $\mathsf{depth}(m_i(i_u, j_u)) > \mathsf{depth}(\mathcal{K})$ for every $u \in \{0, 1, 2\}$;
- $q_{i_0} = q_{i_1} = q_{i_2}$ and $q_{j_0} = q_{j_1} = q_{j_2}$; and
- $[m'_i(i_0, j_0)]_{\equiv_{i+1}} = [m'_i(i_1, j_1)]_{\equiv_{i+1}} = [m'_i(i_2, j_2)]_{\equiv_{i+1}}$.

First assume $m'_i(i_1, j_1) \neq \varepsilon$. Define $\overline{\pi}$ to be the path obtained from π by removing the subpath in which $m_i(i_0, j_0)$ is read and substituting it by the subpath in which $m_i(i_1, j_1)$ is read, i.e.,

$$\overline{\pi} := q_0(a_0, b_0)q_1 \cdots q_{i_0}(a_{i_1}, b_{i_1})q_{i_1+1} \cdots q_{j_1-1}(a_{j_1-1}, b_{j_1-1})q_{j_0}(a_{j_0}, b_{j_0})q_{j_0+1} \cdots q_r.$$

Obviously, $\overline{\pi}$ is a path in $\mathcal{A}_i$ since $q_{i_0} = q_{i_1}$ and $q_{j_0} = q_{j_1}$. Let $t(v)$ be a term with a variable v occurring exactly once in t such that $m_i = t[v/m_i(i_0, j_0)]$. Thus, the input label of $\overline{\pi}$ is $\overline{m}_i := t[v/m_i(i_1, j_1)] = m_i(0, i_0)m_i(i_1, j_1)m_i(j_0, r)$. Using $\mathsf{depth}(m_i(i_0, j_0)) > \mathsf{depth}(\mathcal{K})$ and $\mathsf{depth}(m_i(i_1, j_1)) > \mathsf{depth}(\mathcal{K})$, we can conclude that $m_i(i_0, j_0)$ and $m_i(i_1, j_1)$ are not submessage of some message in $\mathcal{K}$. From this, it easily follows that $\overline{m}_i \in \mathsf{d}(\mathcal{K})$.

The output label of $\overline{\pi}$ is $\overline{m}'_i := m'_i(0, i_0)m'_i(i_1, j_1)m'_i(j_0, r)$. We know that the messages $m_i(i_0, j_0)$ and $m_i(i_1, j_1)$ are the input labels of the strict subpaths in π from position i_0 to j_0 and i_1 to j_1, respectively. Now, the definition of message transducer guarantees that $m'_i(i_0, j_0)$ and $m'_i(i_1, j_1)$ are messages as well. Consequently, $\overline{m}'_i$, obtained from substituting $m'_i(i_0, j_0)$ in m'_i by $m'_i(i_1, j_1)$, is also a message. (Here we use that $m'_i(i_1, j_1) \neq \varepsilon$. Otherwise, $\overline{m}'_i$ could contain a word of the form $\mathtt{enc}_a() \notin \mathcal{M}_\varepsilon$.) In particular, there exists a term $t'(v)$ with a variable v occurring exactly once in t' such that $m'_i = t'[v/m'(i_0, j_0)]$ and $\overline{m}'_i = t'[v/m'_i(i_1, j_1)]$.

With this and using $m'_i(i_0, j_0) \equiv_{i+1} m'_i(i_1, j_1)$, Lemma 15 implies $m'_i \equiv_{i+1} \overline{m}'_i$. Now, Proposition 17 guarantees that the problem $(\overline{\mathcal{K}}, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$ with $\overline{\mathcal{K}} := \mathcal{K} \cup \{\overline{m}'_i\}$ has a solution $(\overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$. Thus, $(\overline{m}_i, \overline{m}'_i, \overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_i, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$.

Finally, let us consider the case $m'_i(i_1, j_1) = \varepsilon$. Since $m'_i(i_2, j_2)$ is a submessage of $m'_i(i_1, j_1)$, it follows $m'_i(i_2, j_2) = \varepsilon$. Define $\overline{m}_i := m_i(0, i_1)m_i(i_2, j_2)m(j_1, r)$ and $\overline{m}'_i := m'_i(0, i_1)m'_i(i_2, j_2)m'(j_1, r)$. It follows, $\overline{m}'_i = m'_i$. Thus, $(\overline{m}_i, \overline{m}'_i, m_{i+1}, m'_{i+1}, \ldots, m_{k-1}, m'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_i, \mathcal{A}_{i+1}, \ldots, \mathcal{A}_{k-1})$.

Iterating this argument, Proposition 20 follows.

6.3 Bounding the Depth of Output Labels

In what follows, let $\mathcal{A}$ be a message transducer and π be a path in $\mathcal{A}$ of the form

$$q_0(a_0, b_0)q_1(a_1, b_1) \cdots (a_{r-1}, b_{r-1})q_r \tag{3}$$

with $r > 0$ and $a_i, b_i \in \Sigma_{\mathcal{N}} \cup \{\varepsilon\}$ for every $i < r$ such that $a_0 \cdots a_{r-1},\ b_0 \cdots b_{r-1} \in \mathcal{M}_\varepsilon$. Let l_π and l'_π be the input and output level function of π, respectively (cf. Section 6.2).

The following proposition says that at any position in π, the level of the output at this position is bounded by the level of the input.

Proposition 21 *Let $\mathcal{A} = (Q, \Sigma_{\mathcal{N}}, \{q_I\}, \Delta, \{q_F\})$ be a message transducer, $n := |Q|$, and π be a path in $\mathcal{A}$ of the form (3) such that $q_0 = q_I$ and $q_r = q_F$, or π is strict, i.e., $\pi \in_s \mathcal{A}$. Then, it follows $l'_\pi(i) \leq (n^2 \cdot (2n+1)+1) \cdot (l_\pi(i)+1)$ for every $i \leq r$.*

We define

$$\mathsf{depth}(\mathcal{A}) := n^2 \cdot (2n+1) + 1.$$

As a corollary, we obtain that the depth of the output of a message transducer is bounded by the depth of the input.

Corollary 22 *Let $\mathcal{A} = (Q, \Sigma_{\mathcal{N}}, I, \Delta, F)$ be a message transducer. Then, for every $m, m' \in \mathcal{M}_\varepsilon$ with $m' \in \mathcal{A}(m)$, or $p(m, m')q \in_s \mathcal{A}$, for $p, q \in Q$: $\mathsf{depth}(m') \leq \mathsf{depth}(\mathcal{A}) \cdot (\mathsf{depth}(m)+1)$.*

Together with Proposition 29, this yields:

Corollary 23 *Assume that $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ is a solvable instance of the path problem. Then, there exists a solution $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$ of this instance such that the depth of m_i and m'_i is bounded in the size of the instance. More precisely, $\mathsf{depth}(m_i) \leq 2 \cdot |Q_i|^2 \cdot I(\equiv_{i+1}) + \mathsf{depth}(\mathcal{K}) + 2$ and $\mathsf{depth}(m'_i) \leq \mathsf{depth}(\mathcal{A}_i) \cdot (\mathsf{depth}(m_i)+1)$.*

By induction, we can deduce that the depth of all messages in a solution of an instance of the path problem can be bounded.

The proof of Proposition 21 uses that the length of paths in $\mathcal{A}$ can be restricted by $\mathsf{depth}(\mathcal{A})$. To show this we cannot simple use the usual pumping argument on finite automata since if we truncate a path, we want to guarantee that the input label of the resulting path is still a message. Therefore, the path can only be cut at certain positions. One possibility is that the path is cut such that an input label of the form $\mathtt{enc}_a(\cdots \mathtt{enc}_b(\cdot) \cdots)$ is replaced by $\mathtt{enc}_b(\cdot)$. Alternatively, one can cut a path such that if the input label is of the form $xw\gamma$ it is replaced by $x\gamma$, where x and xw are left half-messages of the same level (and thus, γ is a right half-message of this level). A little technical problem in this case is that $x\gamma$ may not be a message since it can contain a word of the form $\mathtt{enc}_a()$, which is not a message. The following lemma, shows how to solve this problem. For a message $m = c_0 \cdots c_{r-1} \in \mathcal{M}$ with $c_i \in \Sigma_{\mathcal{N}}$, $i < k$, let $l_m(i) := l(c_0 \cdots c_{i-1})$ for $i \leq r$.

Lemma 24 *Let $m = c_0 \cdots c_{r-1} \in \mathcal{M}$ be a message with $c_i \in \Sigma_{\mathcal{N}}$, for all $i < r$.*

1. *If for $0 \leq i < j \leq r$, $l_m(i) = l_m(j) = 0$, then $c_0 \cdots c_{i-1} c_j \cdots c_{r-1} \in \mathcal{M}_\varepsilon$.*

2. *Let $0 < i < j < r$ with $l_m(i) = l_m(j)$. Then, $c_{i-1} \neq \mathtt{enc}_a($, for any $a \in \mathcal{N}$, or $c_j \neq)$ iff $c_0 \cdots c_{i-1}c_j \cdots c_{r-1} \in \mathcal{M}_\varepsilon$.*

3. *If $0 < i_0 < i_1 < i_2 < r$ with $l_m(i_0) = l_m(i_1) = l_m(i_2)$, then $c_0 \cdots c_{i_0-1}c_{i_1} \cdots c_{r-1} \in \mathcal{M}_\varepsilon$ or $c_0 \cdots c_{i_1-1}c_{i_2} \cdots c_{r-1} \in \mathcal{M}_\varepsilon$.*

PROOF. The first statement is easy to see. For the second statement, we note that the condition $c_{i-1} \neq \mathtt{enc}_a($, for any $a \in \mathcal{N}$, or $c_j \neq)$ guarantees that $c_{i-1}c_j$ is not of the form $\mathtt{enc}_b()$, for some $b \in \mathcal{N}$. Having ruled out this possibility, it is not hard to show that $c_0 \cdots c_{i-1}c_j \cdots c_{r-1}$ is a message. Conversely, if $c_0 \cdots c_{i-1}c_j \cdots c_{r-1}$ is a message, then, since it does not contain a submessage of the form $\mathtt{enc}_b()$, it follows $c_{i-1} \neq \mathtt{enc}_a($, for any $a \in \mathcal{N}$, or $c_j \neq)$.

For the third statement, assume that neither $c_0 \cdots c_{i_0-1}c_{i_1} \cdots c_{r-1} \in \mathcal{M}_\varepsilon$ nor $c_0 \cdots c_{i_1-1}c_{i_2} \cdots c_{r-1} \in \mathcal{M}_\varepsilon$. From 2. it follows: $c_{i_0} = \mathtt{enc}_a($, for some $a \in \mathcal{N}$, $c_{i_1} =)$, $c_{i_1-1} = \mathtt{enc}_b($, for some $b \in \mathcal{N}$, and $c_{i_2} =)$. But then c contains as a submessage $c_{i_1-1}c_{i_1} = \mathtt{enc}_b()$, in contradiction to the fact that c is a message. □

Now, we show how to bound the length of paths by $\mathsf{depth}(\mathcal{A})$.

Lemma 25 *Let π be a path of the form (3) in a message transducer $\mathcal{A}$, and let n be the number of states of $\mathcal{A}$. Then, there exists a path π' from q_0 to q_r in $\mathcal{A}$ such that the length of π' is $< \mathsf{depth}(\mathcal{A})$ and the input label of π' is a message. Moreover, if $a_{r-1} \neq \varepsilon$, then the input label of the last transition in π' is distinct from ε.*

PROOF. Let $m := a_0 \cdots a_{r-1}$. We first show that we can restrict the depth of an input label of a path from q_0 to q_r by n^2.

Assume that $\mathsf{depth}(m) > n^2$. Then, there exist $i_0 < j_0 < j_1 < i_1 \leq r$ such that $q_{i_0} = q_{j_0}$, $q_{j_1} = q_{i_1}$, $a_{i_0} = \mathtt{enc}_a($, for some $a \in \mathcal{N}$, $a_{i_1-1} =)$ (the corresponding closing parenthesis to a_{i_0}), and analogously, $a_{j_0} = \mathtt{enc}_b($, for some $b \in \mathcal{N}$, and $a_{j_1-1} =)$. It follows that the path π' given as

$$q_0(a_0, b_0)q_1 \cdots q_{i_0}(a_{j_0}, b_{j_0})q_{j_0+1} \cdots (a_{j_1-1}, b_{j_1-1})q_{j_1}(a_{i_1}, b_{i_1})q_{i_1+1} \cdots q_r$$

is also a path in $\mathcal{A}$ from q_0 to q_r such that its input label is a message. Note that the input label of the last transition of π and the one of π' coincide. Iterating this argument, we obtain a path from q_0 to q_r such that the input label is a message of depth $\leq n^2$.

Thus, from now on we may assume that $\mathsf{depth}(m) \leq n^2$. In particular, for $l_\pi(i)$, as defined above, we know $l_\pi(i) \leq n^2$ for every $i \leq r$,. Now assume $r \geq \mathsf{depth}(\mathcal{A})$. Then, there must exist an $l \leq n^2$ such that $l_\pi(i) = l$ for $> 2n + 1$ many $i \leq r$. Thus, there exist $0 \leq i_0 < i_1 < i_2 < r$ such that $q_{i_0} = q_{i_1} = q_{i_2}$ and $l_\pi(i_0) = l_\pi(i_1) = l_\pi(i_2)$. It follows that for $j \in \{0, 1\}$ the path π'_j given as

$$q_0(a_0, b_0)q_1 \cdots q_{i_j}(a_{i_{j+1}}, b_{i_{j+1}})q_{i_{j+1}+1}(a_{i_{j+1}+1}, b_{i_{j+1}+1}) \cdots q_r$$

is a path in $\mathcal{A}$ from q_0 to q_r. Lemma 24 implies that the input label of π'_0 or π'_1 is a message. Finally, since $i_2 < r$, the last transition in π'_j, for $j \in \{0, 1\}$, coincides with the one for π.

Iterating this construction, we obtain a path π' with the desired properties. □

Proof of Proposition 21. Let π be a path as given in Proposition 21, and assume there exists $i \leq r$ such that $l'_\pi(i) > \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(i)+1)$. We may assume that $l_\pi(i)$ is minimal, i.e., there exists no $j \leq r$ such that $l_\pi(j) < l_\pi(i)$ and $l'_\pi(j) > \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(j)+1)$. We distinguish two cases.

$l_\pi(i) > 0$. Let $j > i$ be minimal with $l_\pi(j) = l_\pi(i) - 1$. From the minimality of $l_\pi(i)$ it follows $l'_\pi(j) \leq \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(j)+1)$. But then, there must exist $> \mathsf{depth}(\mathcal{A})$ many positions s with $i \leq s \leq j-1$ and $b_s =)$. (Otherwise, $l'_\pi(j) = l'_\pi(i) - g$ for some $g \leq \mathsf{depth}(\mathcal{A})$. Thus, $l'_\pi(j) > \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(i)+1) - \mathsf{depth}(\mathcal{A}) = \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(j)+1)$, in contradiction to the minimality of $l_\pi(i)$.)

By the choice of j, we know that the word $a_i \cdots a_{j-2}$ is a message and that it is the input label of the subpath π' of π from q_i to q_{j-1}. (Note that $a_{j-1} =)$.) Lemma 25 implies that there is also a path π'' in $\mathcal{A}$ from q_i to q_{j-1} of length $< \mathsf{depth}(\mathcal{A})$ such that the input label of π'' is a message. Replacing π' in π by π'' yields a new, shorter path, say $\overline{\pi}$, from q_0 to q_r, with the properties required in the proposition. In particular, the input label of $\overline{\pi}$ is a message. Let $l_{\overline{\pi}}(j)$ and $l'_{\overline{\pi}}(j)$ be the input and output level functions of $\overline{\pi}$, respectively. Moreover, let $\overline{j}$ denote the position in $\overline{\pi}$ corresponding to j in π. Then, we have $l'_{\overline{\pi}}(\overline{j}) > \mathsf{depth}(\mathcal{A}) \cdot (l_{\overline{\pi}}(\overline{j})+1)$ and $l_{\overline{\pi}}(\overline{j}) < l_{\overline{\pi}}(i)$, since $< \mathsf{depth}(\mathcal{A})$ parenthesis have been closed between position i and $\overline{j}-1$ in $\overline{\pi}$, i.e., $\leq \mathsf{depth}(\mathcal{A})$ parenthesis between i and $\overline{j}$. Iterating this argument shows that there exists a path π with the desired properties such that $l_\pi(i)$, as chosen above, is 0. Thus, the case $l_\pi(i) = 0$ applies. (However, we will see that this case leads to a contradiction.)

$l_\pi(i) = 0$. Let π' denote the path from q_i to q_r with input label $a_i \cdots a_{r-1}$ and output label $b_i \cdots b_{r-1}$. Since $l'_\pi(i) > \mathsf{depth}(\mathcal{A}) \cdot (l_\pi(i)+1)$, the output label must contain $> \mathsf{depth}(\mathcal{A})$ closing parentheses. However, according to Lemma 25, there exists a path π'' from q_i to q_r in $\mathcal{A}$ such that the input label of π'' is a message and the length of π'' is $< \mathsf{depth}(\mathcal{A})$. Then, replacing π' in π by π'' yields a new path $\overline{\pi}$ from q_0 to q_r such that the input label is a message. Moreover, if π is strict, then the new path is also strict: if the last transition in π' ended with a input label distinct from ε, then, according to Lemma 25, this can be achieved for π'' as well.

Now, since the input label of $\overline{\pi}$ is a message, the conditions on message-transducers imply that the output label must also be a message. However, this is not true, since at least two closing parenthesis are missing.

6.4 The Path Truncation Ordering $\preceq_i^l$

We now extend $\preceq_i$ to the path truncation ordering $\preceq_i^l$ on half-messages and show that $\preceq_i^l$ is compatible with right concatenation of right half-messages (Section 6.4.1). We also prove that the index of the equivalence relation corresponding to $\preceq_i^l$ is finite (Section 6.4.2).

For the definition of left and right half-messages and their levels $l(\cdot)$ see Section 6.2. If α is a left half-message, then there exist unique messages $x_0, \ldots, x_{l(\alpha)} \in \mathcal{M}_\varepsilon$, and

$a_1, \ldots, a_{l(\alpha)-1} \in \mathcal{N}$ such that

$$\alpha = x_{l(\alpha)} \mathtt{enc}_{a_{l(\alpha)}}(x_{l(\alpha)-1} \mathtt{enc}_{a_{l(\alpha)-1}}(x_{l(\alpha)-2} \cdots \mathtt{enc}_{a_1}(x_0.$$

We define the *j-level half message of α* to be

$$\alpha_j := \mathtt{enc}_{a_j}(x_{j-1} \mathtt{enc}_{a_{j-1}}(x_{j-2} \cdots \mathtt{enc}_{a_1}(x_0$$

for $1 \leq j \leq l(\alpha)$. Note that $l(\alpha_j) = j$. Moreover, we define α^* to be the message obtained from α by adding the missing closing parentheses, i.e.,

$$\alpha^* := \alpha \underbrace{) \cdots)}_{l(\alpha)}.$$

Finally, let $p(\alpha) := a_{l(\alpha)} \cdots a_1$. In order to define $\preceq_i^l$, we first introduce the ordering $\preceq^l$.

Definition 26 *Let $l \geq 0$ and α, α' be non-empty left half-messages of level l, i.e., $\alpha \neq \varepsilon$, $\alpha' \neq \varepsilon$, and $l(\alpha) = l(\alpha') = l$. Define $\alpha \preceq^l \alpha'$ iff $\alpha^* \preceq \alpha'^*$ and $p(\alpha) = p(\alpha')$.*

Clearly, $\preceq^l$ is a quasi-ordering. For the definition of $\preceq_i^l$ we need some more notation. If α and β are left half-messages, and $p, q \in Q_i$, then $p(\alpha, \beta)q \in_h \mathcal{A}_i$ means that i) $p(\alpha, \beta)q \in_s \mathcal{A}_i$, and ii) there exist right half-messages γ, γ', and a state $q' \in Q_i$ such that $l(\gamma) = l(\alpha)$, $l(\gamma') = l(\beta)$, and $p(\alpha, \beta)q(\gamma, \gamma')q'$ is a strict path in $\mathcal{A}_i$. In other words, the strict path $p(\alpha, \beta)q$ can be extended to a strict path such that the input and output labels are messages.

Definition 27 *For every $l \geq 0$ and $i \leq k$, the* path truncation ordering *$\preceq_i^l$ is defined as follows: for left half-messages α, α' of level l, $\alpha \preceq_i^l \alpha'$ iff i) $\alpha = \alpha' = \varepsilon$, or ii) $\alpha \neq \varepsilon$, $\alpha' \neq \varepsilon$, $\alpha \preceq^l \alpha'$, and if $i < k$, then*

1. *$\alpha \preceq_{i+1}^l \alpha'$;*
2. *$\alpha^* \preceq_i \alpha'^*$;*
3. *$\alpha_j \sqsubseteq_i^j \alpha'_j$ for every $1 \leq j \leq l$;*
4. *if $l \geq 1$ and $x, x' \in \mathcal{M}_\varepsilon$ with $\alpha = x\alpha_l$ and $\alpha' = x'\alpha'_l$, then $x \sqsubseteq_i x'$.*

For $i < k$, $l \geq 0$, left half-messages α, α' of level l, we define $\alpha \sqsubseteq_i^l \alpha'$ iff i) $\alpha = \alpha' = \varepsilon$, or ii) $\alpha \neq \varepsilon$, $\alpha' \neq \varepsilon$, and for every left half-message β and every $p, q \in Q_i$, $p(\alpha, \beta)q \in_h \mathcal{A}_i$ implies that there exists a left half-message β' with $l(\beta') = l(\beta)$ such that $p(\alpha', \beta')q \in_h \mathcal{A}_i$ and $\beta \preceq_{i+1}^{l(\beta)} \beta'$.

6.4.1 Right Concatenation of Right Half-messages

We now show, by induction on i, that $\preceq_i^l$ is compatible with right concatenation of right half-messages. The case $i = k$ is shown in the following lemma.

Lemma 28 *Let α, α' be left half-messages of level $l \geq 0$. Then, $\alpha \preceq^l \alpha'$ implies $\alpha\gamma \preceq \alpha'\gamma$ for every right half-message γ of level l.*

PROOF. Assume $\alpha \preceq^l \alpha'$. If $l = 0$, then α, α', and γ are messages, then the lemma follows from Lemma 14, when we set $t := v_0 v_1$ and consider $t[v_0/\alpha, v_1/\gamma]$ and $t[v_0/\alpha', v_1/\gamma]$. In what follows we assume $l > 0$.

We need to show that $\mathsf{an}_{\mathcal{N}}(\alpha\gamma, N) \subseteq \mathsf{an}_{\mathcal{N}}(\alpha'\gamma, N)$ for every $N \subseteq \mathcal{N}$. To this end, we define two mappings F and F' from $2^{\mathcal{N}}$ into $2^{\mathcal{N}}$, where $2^{\mathcal{N}}$ denotes the powerset of $\mathcal{N}$. For every $N \subseteq \mathcal{N}$,

$$\begin{aligned} F(N) &:= \mathsf{an}_{\mathcal{N}}(\mathtt{enc}_{p(\alpha)}(\gamma), \mathsf{an}_{\mathcal{N}}(\alpha^*, N)) \text{ and} \\ F'(N) &:= \mathsf{an}_{\mathcal{N}}(\mathtt{enc}_{p(\alpha')}(\gamma), \mathsf{an}_{\mathcal{N}}(\alpha'^*, N)), \end{aligned}$$

where $\mathtt{enc}_w(\gamma)$ for some non-empty word $w = a_0 \cdots a_{l-1} \in \mathcal{N}^+$, denotes the message

$$\mathtt{enc}_{a_0}(\mathtt{enc}_{a_1}(\cdots \mathtt{enc}_{a_{l-1}}(\gamma.$$

(Note that the corresponding closing parenthesis to "$\mathtt{enc}_{a_j}($" are contained in γ.)

Because $\alpha \preceq^l \alpha'$, we know that $p(\alpha) = p(\alpha')$ and $\alpha^* \preceq \alpha'^*$. Thus, $\mathsf{an}_{\mathcal{N}}(\alpha^*, N) \subseteq \mathsf{an}_{\mathcal{N}}(\alpha'^*, N)$. Consequently, $F(N) \subseteq F'(N)$. Using similar techniques as in the proof of Lemma 14, it is rather straightforward to show

$$\begin{aligned} \mathsf{an}_{\mathcal{N}}(\alpha\gamma, N) &= \mathsf{lfp}_N(F) := \bigcup_{j \geq 0} F^j(N), \text{ and} \\ \mathsf{an}_{\mathcal{N}}(\alpha'\gamma, N) &= \mathsf{lfp}_N(F'). \end{aligned}$$

From this the lemma follows. □

We can now proof the main proposition of this subsection.

Proposition 29 *Let α, α' be left half-messages of level $l \geq 0$ and let $i \leq k$. Then, $\alpha \preceq^l_i \alpha'$ implies $\alpha\gamma \preceq_i \alpha'\gamma$ for every right half-message γ of level l.*

PROOF. Assume $\alpha \preceq^l_i \alpha'$. If $\alpha = \varepsilon$, then $\alpha' = \varepsilon$, and thus, $\alpha\gamma \preceq_i \alpha'\gamma$. Assume $\alpha \neq \varepsilon$ and $\alpha' \neq \varepsilon$. The rest of the proof is by induction on $i \leq k$. The base case, $i = k$, follows from Lemma 28.

Now assume $i < k$. If $l = 0$, then α, α', and γ are messages, and $\alpha\gamma \preceq_i \alpha'\gamma$ follows from Lemma 15 with $t = v_0 v_1$. Therefore, we may assume that $l > 0$. To prove $\alpha\gamma \preceq_i \alpha'\gamma$, we must show the conditions in Definition 12.

1. By definition, $\alpha \preceq^l_i \alpha'$ implies $\alpha \preceq^l_{i+1} \alpha'$, and thus, with the induction hypothesis, we obtain $\alpha\gamma \preceq_{i+1} \alpha'\gamma$.

2. Let $N \subseteq \mathcal{N}$ and $x \in \mathsf{an}(\alpha\gamma, N)$ with $x = \mathtt{enc}_a(z)$ for some message z and $a \in \mathcal{N}$. We distinguish three cases:

   (a) x is a submessage of γ. According to Lemma 28, $\alpha\gamma \preceq \alpha'\gamma$. Thus, $\mathsf{an}_{\mathcal{N}}(\alpha\gamma, N) \subseteq \mathsf{an}_{\mathcal{N}}(\alpha'\gamma, N)$. Using $p(\alpha) = p(\alpha')$, it follows $x \in \mathsf{an}(\alpha'\gamma, N)$, and we can simply choose $x' := x$.

(b) x is a submessage of α. It follows $x \in \mathsf{an}(\alpha^*, \mathsf{an}_{\mathcal{N}}(\alpha\gamma, N))$. Since $\alpha^* \preceq_i \alpha'^*$, there exists $x' \in \mathsf{an}(\alpha'^*, \mathsf{an}(\alpha\gamma, N))$ such that x' is of the form $\mathtt{enc}_b(z')$ for some message z' and $b \in \mathcal{N}$ and $x \sqsubseteq_i x'$. Finally, because $\mathsf{an}_{\mathcal{N}}(\alpha\gamma, N) \subseteq \mathsf{an}_{\mathcal{N}}(\alpha'\gamma, N)$ (Lemma 28), it follows $x' \in \mathsf{an}(\alpha'^*, \mathsf{an}(\alpha'\gamma, N))$, and thus, $x' \in \mathsf{an}(\alpha'\gamma, N)$.

(c) x is of the form $\alpha_j\gamma'$ for some $1 \leq j \leq l$ and a right half-message γ' such that γ' is a prefix of γ with $l(\gamma') = l(\alpha_j)$. Define $x' := \alpha'_j\gamma'$. Obviously, x' is a message of the form $\mathtt{enc}_a(z')$ for some message z' and $a \in \mathcal{N}$. Since $\mathsf{an}_{\mathcal{N}}(\alpha\gamma, N) \subseteq \mathsf{an}_{\mathcal{N}}(\alpha'\gamma, N)$ (Lemma 28) and $p(\alpha) = p(\alpha')$ it follows $x' \in \mathsf{an}(\alpha'\gamma, N)$. We need to show $x \sqsubseteq_i x'$.

Let $p, q \in Q_i$, $y \in \mathcal{M}_\varepsilon$ such that $p(x, y)q \in_s \mathcal{A}_i$. There exists $p' \in Q_i$, and a left half-message β and a right half-message β' such that $y = \beta\beta'$, $l(\beta) = l(\beta')$, $p(\alpha_j, \beta)p' \in_s \mathcal{A}_i$, and $p'(\gamma', \beta')q \in \mathcal{A}_i$. We know that $p(\alpha_j, \beta)p'(\gamma', \beta')q$ is a strict path in $\mathcal{A}_i$ and that $x = \alpha_j\gamma'$ and $y = \beta\beta'$ are messages. Thus, $p(\alpha_j, \beta)p' \in_h \mathcal{A}_i$. From $\alpha \preceq_i^l \alpha'$, we obtain $\alpha_j \sqsubseteq_i^{l(\alpha_j)} \alpha'_j$, and consequently, there exists a left half-message β'' with $l(\beta) = l(\beta'')$, $p(\alpha'_j, \beta')p' \in_s \mathcal{A}_i$, and $\beta \preceq_{i+1}^{l(\beta)} \beta''$. This yields that $p(\alpha'_j, \beta')p'(\gamma', \beta')q$ is a strict path from p to q in $\mathcal{A}_i$ with input label x' and output label $y' := \beta''\beta'$. By the induction hypothesis, $y = \beta\beta' \preceq_{i+1} \beta''\beta' = y'$.

3. We show $\alpha\gamma \sqsubseteq_i \alpha'\gamma$. Let $p, q \in Q_i$ and $y \in \mathcal{M}_\varepsilon$ with $p(\alpha\gamma, y)q \in_s \mathcal{A}_i$. Since $l > 0$, α has the form $x\alpha_l$ for some message x. We first assume $x \neq \varepsilon$. Thus, there exist words $y_0, y_1, \beta, \beta' \in \Sigma_{\mathcal{N}}^*$ and states p_0, p_1, p_2 with $p(x, y_0)p_0 \in_s \mathcal{A}_i$, $p_0(\varepsilon, y_1)p_1 \in \mathcal{A}_i$, $p_1(\alpha_l, \beta)p_2 \in_s \mathcal{A}_i$, and $p_2(\gamma, \beta')q \in A_i$, where the input label of the last transition of the latter path is $\neq \varepsilon$.

   Since the first path is strict and x is a message, by the definition of message transducer, it follows $y_0 \in \mathcal{M}_\varepsilon$. Since the path from p_1 to q is strict and $\alpha_l\gamma$ is a message, we know that $\beta\beta'$ is a message. Particularly, β is a left half-message and β' is a right half-message with $l(\beta) = l(\beta')$. Finally, since $y = y_0y_1\beta\beta' \in \mathcal{M}_\varepsilon$ and y_0, $\beta\beta' \in \mathcal{M}_\varepsilon$, we can conclude $y_1 \in \mathcal{M}_\varepsilon$.

   Now, $\alpha \preceq_i^l \alpha'$ implies $\alpha_l \sqsubseteq_i^l \alpha'_l$, and we know that $p_1(\alpha_l, \beta)p_2(\gamma, \beta')q$ is a strict path in $\mathcal{A}_i$, and $\alpha_l\gamma$ and $\beta\beta'$ are messages. Thus, $p_1(\alpha_l, \beta)p_2 \in_h \mathcal{A}_i$. As a result, there exists a left half-message β'' with $l(\beta'') = l(\beta)$, $p_1(\alpha'_l, \beta'')p_2 \in_s \mathcal{A}_i$, and $\beta \preceq_{i+1}^{l(\beta)} \beta''$. Moreover, if $\alpha' = x'\alpha'_l$, then $\alpha \preceq_i^l \alpha'$ implies $x \sqsubseteq_i x'$. Thus, there exists $y'_0 \in \mathcal{M}_\varepsilon$ with $p(x', y'_0), p_0 \in_s \mathcal{A}_i$ and $y_0 \preceq_{i+1} y'_0$. Therefore, replacing in the path $p(x, y)q$ the subpath $p(x, y_0)p_0$ by $p(x', y'_0), p_0$ and $p_1(\alpha_l, \beta)p_2$ by $p_1(\alpha'_l, \beta'')p_2$ yields a strict path from p to q with input label $\alpha'\gamma$ and output label $y' := y'_0y_1\beta''\beta' \in \mathcal{M}_\varepsilon$.

   It remains to show $y \preceq_{i+1} y'$. By induction, $\beta \preceq_{i+1}^{l(\beta)} \beta''$ implies $\beta\beta' \preceq_{i+1} \beta\beta''$. We also know $y_0 \preceq_{i+1} y'_0$ and $y_1 \preceq_{i+1} y_1$. Thus, by Lemma 15, with $t = v_0v_1v_2$ we obtain $y = t[v_0/y_0, v_1/y_1, v_2/\beta\beta'] \preceq_{i+1} t[v_0/y'_0, v_1/y_1, v_2/\beta''\beta'] = y'$.

   If $x = \varepsilon$ and x' is defined as above, it follows $x' = \varepsilon$, because $x \sqsubseteq_i x'$. The rest of the proof is similar to the case $x \neq \varepsilon$. □

6.4.2 The Index of the Path Truncation Ordering

Let $\equiv^l$, $\equiv^l_i$, and $=^l_i$ denote the equivalence relations corresponding to $\preceq^l$, $\preceq^l_i$, and $\sqsubseteq^l_i$.

We prove that $\equiv^l_i$ has finite index by induction on i. The base case follows from the following lemma.

Lemma 30 *For every $l \geq 0$, the index of $\equiv^l$ is finite, and can be bounded as follows:*

$$I(\equiv^l) \in \mathcal{O}(I(\equiv) \cdot |\mathcal{N}|^l)$$

Proof. Let f_l be a mapping that takes a left half-message α of level l to the tuple $([\alpha^*]_\equiv, p(\alpha))$. Since $\equiv$ has finite index and there are only a finite number of words over $\mathcal{N}$ of length l, the range of f_l is finite, namely, $I(\equiv) \cdot |\mathcal{N}|^l$. Moreover, it is easy to see that $f_l(\alpha) = f_l(\alpha')$ implies $\alpha \equiv^l \alpha'$ for all left half-messages α and α' of level l. $\square$

The proof of the following proposition is very similar to the one for Proposition 19. However, it requires Proposition 21.

Proposition 31 *For all $l \geq 0$, the index of $\equiv^l_k$, and for every $i < k$, the index of $\equiv^l_i$ and $=^l_i$ is finite, and can be bounded as follows:*

$$\begin{aligned} I(=^l_i) &\in \mathcal{O}\left(\left(I(\equiv^{\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}_{i+1}) \cdot 2^{|Q_i|^2}\right)^{\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}\right) \\ I(\equiv^l_i) &\in \mathcal{O}\left(I(\equiv^l_{i+1}) \cdot I(\equiv_i) \cdot I(=^l_i)^l \cdot I(=_i)\right) \end{aligned}$$

Proof. The proof is by induction on i. For $i = k$, Lemma 30 shows that the index of $\equiv^l_k$ is finite. Assume that $i < k$ and the index of $\equiv^l_{i+1}$ is finite. We first show that the index of $=^l_i$ is finite and from this conclude that $\equiv^l_i$ has finite index.

For every $l, l' \geq 0$, we introduce a new equivalence relation on tuples (α, β) with left half-messages α and β. More precisely, for left half-messages $\alpha, \alpha', \beta, \beta'$ with $l(\alpha) = l(\alpha') = l$ and $l(\beta) = l(\beta') = l'$ we define: $(\alpha, \beta) =^{l,l'}_i (\alpha', \beta')$ iff

- $\beta \equiv^{l'}_{i+1} \beta'$, and
- $p(\alpha, \beta)q \in_h \mathcal{A}_i$ iff $p(\alpha', \beta')q \in_h \mathcal{A}_i$, for every $p, q \in Q_i$.

Just as for $=^t_i$ in the proof of Proposition 19, one shows that $=^{l,l'}_i$ has finite index, and that

$$I(=^{l,l'}_i) \in \mathcal{O}(I(\equiv^{l'}_{i+1}) \cdot 2^{|Q_i|^2}).$$

To show that $=^l_i$ has finite index, define for $l, l' \geq 0$ and a left half-message α of level l the set

$$M^{l,l'}_{i,\alpha} := \{[(\alpha, \beta)]_{=^{l,l'}_i} \mid \beta \text{ is a left half-message of level } l'\}$$

and the tuple

$$M^l_{i,\alpha} := (M^{l,l'}_{i,\alpha} \mid l' \leq \mathsf{depth}(\mathcal{A}_i) \cdot (l+1)).$$

Clearly, for fixed l, l', and i the number of different sets $M^{l,l'}_{i,\alpha}$ is finite. Consequently, the set of tuples $M^l_{i,\alpha}$ for fixed l and i is also finite. Together with the following claim, this implies that $=^l_i$ has finite index.

Claim. For all $l \geq 0$ and left half-messages α, α' with $\alpha \neq \varepsilon$, $\alpha' \neq \varepsilon$, and $l(\alpha) = l(\alpha') = l$: $M^l_{i,\alpha} = M^l_{i,\alpha'}$ implies $\alpha =^l_i \alpha'$.

Proof of the claim. Assume $M^l_{i,\alpha} = M^l_{i,\alpha'}$. We show $\alpha \sqsubseteq^l_i \alpha'$; $\alpha' \sqsubseteq^l_i \alpha$ follows by symmetry. Let $p, q \in Q_i$ and β be a left half-message with $p(\alpha,\beta)q \in_h \mathcal{A}_i$. By definition of $\in_h$, their exist right half-messages γ, γ', and a state $q' \in Q_i$ such that $p(\alpha,\beta)q(\gamma,\gamma')q'$ is a strict path in $\mathcal{A}_i$. Then, Proposition 21 implies $l' := l(\beta) \leq \mathsf{depth}(\mathcal{A}_i) \cdot (l+1)$. Now, using $M^l_{i,\alpha} = M^l_{i,\alpha'}$, it follows $[(\alpha,\beta)]_{=^{l,l'}_i} \in M^{l,l'}_{i,\alpha} = M^{l,l'}_{i,\alpha'}$. Thus, there exists a left half-message β' with $l(\beta') = l'$ such that $[(\alpha,\beta)]_{=^{l,l'}_i} = [(\alpha',\beta')]_{=^{l,l'}_i}$. In particular, $\beta \equiv^{l'}_{i+1} \beta'$, and from $p(\alpha,\beta)q \in_h \mathcal{A}_i$ it follows $p(\alpha',\beta')q \in_h \mathcal{A}_i$. This concludes the proof of the claim.

It is easy to see that $I(\equiv^r_i) \leq I(\equiv^{r'}_i)$ and $I(=^r_i) \leq I(=^{r'}_i)$, for every $r \leq r'$. From this we can conclude

$$\begin{aligned} I(=^l_i) &\in \mathcal{O}\left(I\left(=^{l,\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}_i\right)^{\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}\right) \\ &\subseteq \mathcal{O}\left(\left(I\left(\equiv^{\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}_{i+1}\right)\cdot 2^{|Q_i|^2}\right)^{\mathsf{depth}(\mathcal{A}_i)\cdot(l+1)}\right) \end{aligned}$$

Now it is easy to show that the index of $\equiv^l_i$ is finite. Consider the mapping f^l_i that takes a left half-message α of level l to the tuple

$$([\alpha]_{\equiv^l_{i+1}}, [\alpha^*]_{\equiv_i}, [\alpha_1]_{=^1_i}, \ldots, [\alpha_l]_{=^l_i}, [x]_{=_i}),$$

where x is a message such that $\alpha = x\alpha_l$. Obviously, the range of f^l_i is finite. Finally, it is straightforward to prove for left half-messages α, α' with $l(\alpha) = l(\alpha') = l$ that $f^l_i(\alpha) = f^l_i(\alpha')$ implies $\alpha \equiv^l_i \alpha'$. From this, the bound on $I(\equiv^l_i)$ claimed in the proposition follows immediately. □

6.5 Proof of Theorem 10

Putting everything together, we now show that the path problem is decidable. This will conclude the proof of Theorem 10.

We show that to find a solution of an instance of PathProblem, it suffices to consider paths (and thus, messages) of length restricted in the size of the problem instance. To this end, we assume that the instance $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$ has the solution $(m_i, m'_i, \ldots, m_{k-1}, m'_{k-1})$, and then construct a solution with short paths. We do so by induction on $i \leq k$. For $i = k$, nothing is to be shown. For the induction step we need some notation.

Assume

$$\pi = q_0(a_0, b_0)q_1(a_1, b_1)\cdots(a_{r-1}, b_{r-1})q_r \tag{4}$$

is a path in $\mathcal{A}_i$ with $a_0 \cdots a_{r-1} = m_i$, $b_0 \cdots b_{r-1} = m'_i$, $q_0 = q^I_i$, and $q_r = q^F_i$. Let $m_i(j,j') := a_j \cdots a_{j'-1}$ and $m'_i(j,j') := b_j \cdots b_{j'-1}$ for all $0 \leq j \leq j' \leq r$. We define

$l_j := l_\pi(j)$ and $l'_j := l'_\pi(j)$, where l_π and l'_π are the input and output level functions of π (cf. Section 6.2).

Due to Corollary 23, we may assume that the depth of m_i and m'_i is bounded in the size of the problem instance.

We define $N := \mathsf{an}_{\mathcal{N}}(\mathcal{K})$ and $M := \{\mathsf{enc}_a(z) \in \mathsf{an}(\mathcal{K}) \mid z \in \mathcal{M} \text{ and } a \in \mathcal{N}\}$. Similar to the proof of Proposition 17, let n be the number of times a message in M was used to derive $m_i \in \mathsf{d}(\mathcal{K})$ from $\mathcal{K}$, and let $\{x_0, \ldots, x_{n-1}\}$ be the multiset of these messages; an $x_j \in M$ occurs in this multiset as many times as it was used in the derivation of m_i. Also, let $v_0, \ldots, v_{n-1}$ be pairwise distinct variables. Then, there exists a term $t \in \mathsf{d}(N \cup \{v_0, \ldots, v_{n-1}\})$, such that every variable v_j, $j < n$, occurs exactly once in t and $m_i = t[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$. Moreover, there exist positions $i_j, i'_j \leq r$ in π such that $x_j = a_{i_j} a_{i_j+1} \cdots a_{i'_j-1}$, $a_{i_j} \neq \varepsilon$, and $a_{i'_j-1} \neq \varepsilon$, for every $j < n$. That is, the subpath in π between the positions i_j and i'_j is a strict path in $\mathcal{A}_i$ with input label x_j.

Finally, we define a mapping f_π as follows: For every $j \leq r$,

$$f_\pi(j) := \begin{cases} (q_j, l_j, l'_j, [m'_i(0,j)]_{\equiv^{l'_j}_{i+1}}, x_s, m_i(i_s, j)), & \text{there exists } s < n \text{ s.t. } i_s < j < i'_s; \\ (q_j, l_j, l'_j, [m'_i(0,j)]_{\equiv^{l'_j}_{i+1}}), & \text{otherwise.} \end{cases}$$

This mapping indicates at which positions π can be truncated. It distinguishes between positions "inside" and "outside" an x_j. The following lemma makes this precise.

Lemma 32 *If there exist j_0, j_1, and j_2 with $0 \leq j_0 < j_1 < j_2 \leq r$ and $f_\pi(j_0) = f_\pi(j_1) = f_\pi(j_2)$, then there exists $u \in \{0,1\}$ and a solution $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ of $(\mathcal{K}, \mathcal{A}_i, \ldots, A_{k-1})$ such that the path $\overline{\pi}$:=*

$$q_0(a_0, b_0)q_1 \cdots q_{j_u}(a_{j_{u+1}}, b_{j_{u+1}})q_{j_{u+1}+1}(a_{j_{u+1}+1}, b_{j_{u+1}+1}) \cdots (a_{r-1}, b_{r-1})q_r$$

is a path in $\mathcal{A}_i$ from q_i^I to q_i^F with input label $\overline{m}_i = m_i(0, j_u)m_i(j_{u+1}, r)$ and output label $\overline{m}'_i = m'_i(0, j_u)m'_i(j_{u+1}, r)$;

Proof. From $f_\pi(j_0) = f_\pi(j_1) = f_\pi(j_2)$ it follows that $l_{j_0} = l_{j_1} = l_{j_2}$. Now, Lemma 24 implies that there exists $u \in \{0,1\}$ such that $\overline{m}_i := m_i(0, j_u)m_i(j_{u+1}, r)$ is a message. Since $\overline{\pi}$ is a path from q_i^I to q_i^F with input label $\overline{m}_i$, from the properties of message transducers (cf. Definition 8) it follows that $\overline{m}'_i := m'_i(0, j_u)m'_0(j_{u+1}, r)$ must be a message.

Obviously, $\overline{\pi}$ is a path from q_i^I to q_i^F with input label $\overline{m}_i$ and output label $\overline{m}'_i$.

It remains to show that there exist messages $\overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1}$ such that $(\overline{m}_i, \overline{m}'_i, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_i, \ldots, A_{k-1})$.

We first show $\overline{m}_i \in \mathsf{d}(\mathcal{K})$: Note that if $i_s < j_u < i'_s$ for some $s < n$, then $f_\pi(j_u) = f_\pi(j_{u+1})$ implies that there exists $s' < n$ with $x_s = x_{s'}$ and $m_i(i_s, j_u) = m_i(i_{s'}, j_{u+1})$. Thus, $x_s = m_i(i_s, j_u)m_i(j_{u+1}, i_{s'})$. That is, after removing the subpath in π between j_u and j_{u+1}, we still have x_s as a submessage. In this way, the two extra components in the first tuple of the definition of f_π prohibit that only part of an x_s is removed when going from π to π'. From this, it is easy to conclude that there exists a term $\bar{t} \in \mathsf{d}(N \cup \{v_0, \ldots, v_{n-1}\})$, in which every variable v_j, $j < n$, occurs at most once, such that $\overline{m}_i := \bar{t}[v_0/x_0, \ldots, v_{n-1}/x_{n-1}]$. Thus, $\overline{m}_i \in \mathsf{d}(\mathcal{K})$.

Because $f_\pi(j_u) = f_\pi(j_{u+1})$, we know $[m'_i(0, j_u)]_{\equiv^{l'}_{i+1}} = [m'_i(0, j_{u+1})]_{\equiv^{l'}_{i+1}}$ and $l' := l'_{j_u} = l'_{j_{u+1}}$. With Proposition 29 we can conclude

$$m'_i = m'_i(0, j_{u+1})m'_i(j_{u+1}, r) \equiv_{i+1} m'_i(0, j_u)m'_i(j_{u+1}, r) = \overline{m}'_i.$$

Now, Proposition 17 guarantees that the instance $(\overline{\mathcal{K}}, \mathcal{A}_{i+1}, \ldots, A_{k-1})$ with $\overline{\mathcal{K}} := \mathcal{K} \cup \{\overline{m}'_i\}$ has a solution $(\overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$, and thus, $(\overline{m}_i, \overline{m}'_i, \overline{m}_{i+1}, \overline{m}'_{i+1}, \ldots, \overline{m}_{k-1}, \overline{m}'_{k-1})$ is a solution of $(\mathcal{K}, \mathcal{A}_i, \ldots, \mathcal{A}_{k-1})$. □

We now need to show that the range of f_π can be bounded in the size of the problem instance. By assumption, the depth of m_i and m'_i is bounded, and thus, l_j and l'_j are bounded for every $j \leq r$ in the size of the problem instance. Now, by Proposition 31, it easily follows that the range of f_π is bounded in the size of the problem instance. Consequently, applying Lemma 32, the length of the paths $q_i^I(m_i, m'_i)q_i^F \in \mathcal{A}_i$ can be bounded as well (and this bound can be computed effectively). By induction, this holds for every path $q_j^I(m_j, m'_j)q_j^F \in \mathcal{A}_j$, $i \leq j < k$. Thus, given an instance $(\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$ of the path problem, a decision algorithm can first compute the bound on the length of the paths and then, enumerate all paths of length restricted by the (computed) bound and check whether their labels satisfy the required conditions. We conclude:

Proposition 33 PathProblem *is decidable.*

As an immediate consequence, we obtain Theorem 10.

7 A Complexity Lower Bound

We prove the following theorem.

Theorem 34 *For transducer-based protocols,* Attack *is PSPACE-hard.*

We first show:

Theorem 35 PathProblem *is PSPACE-hard.*

This is done by reduction from the finite automata intersection problem, which has been shown to be PSPACE-complete by Kozen (see [11]).

The *finite automata intersection problem* is defined as follows: Given $k \geq 0$ deterministic finite automata $\mathcal{B}_0, \ldots, \mathcal{B}_{k-1}$ with a common alphabet Σ, decide whether there exists a word $w \in \Sigma^*$ accepted by $\mathcal{B}_i$ for all $i < k$.[7]

In what follows, assume $\mathcal{B}_i = (Q_i, \Sigma, q_i^I, \delta_i, F_i)$, where Q_i is the set of states, q_i^I is the initial state, δ_i is the transition function, and F_i is the set of final states of $\mathcal{B}_i$.

Given the $\mathcal{B}_i$'s, we define the corresponding path problem as follows. The set of atomic messages $\mathcal{N}$ is $\Sigma \cup \{a, \mathsf{secret}\}$, where a and secret are new atomic messages. The initial intruder knowledge $\mathcal{K}$ is Σ. The message transducer $\mathcal{A}_i$ will correspond to $\mathcal{B}_i$. Before $\mathcal{A}_0$ reads a letter, it outputs "$\mathtt{enc}_a($", then simulating $\mathcal{B}_0$, reads some word

[7] If k is fixed, this problem can be decided in polynomial time.

w and also outputs this words, and finally outputs ")". In other words, $\mathcal{A}_0$ returns messages of the form $\mathtt{enc}_a(w)$, where w is accepted by $\mathcal{B}_0$. The message transducers $\mathcal{A}_i$, $0 < i < k$, do the same, but they accept the input messages to be of the form $\mathtt{enc}_a(w)$, and they only output this message if w is accepted by $\mathcal{B}_i$. Additionally, $\mathcal{A}_{n-1}$ returns secret. The encryption of w guarantees that the intruder must send the same word w ($\mathtt{enc}_a(w)$) to all the $\mathcal{A}_i$'s. Without encryption, the intruder could send different words w' to every $\mathcal{A}_i$. Formally,

$$\mathcal{A}_0 := (Q_0 \cup \{q_0^0, q_0^1, q_0^2, q_0^3\}, \Sigma_{\mathcal{N}}, \{q_0^0\}, \Delta_0, \{q_0^3\})$$

with

$$\begin{array}{rcl}\Delta_0 & := & \{(q_0^0, \varepsilon, \mathtt{enc}_a(, q_0^1),\ (q_0^1, \varepsilon, \varepsilon, q_0^I)\} \cup \\ & & \{(q, b, b, q') \mid \delta_0(q, b) = q'\} \cup \\ & & \{(q, \varepsilon, \varepsilon, q_0^2) \mid q \in F_0\} \cup \\ & & \{(q_0^2, \varepsilon,), q_0^3)\},\end{array}$$

and for $0 < i < k-1$,

$$\mathcal{A}_i := (Q_i \cup \{q_i^0, q_i^1, q_i^2, q_i^3\}, \Sigma_{\mathcal{N}}, \{q_i^0\}, \Delta_i, \{q_i^3\})$$

with

$$\begin{array}{rcl}\Delta_i & := & \{(q_i^0, \mathtt{enc}_a(, \mathtt{enc}_a(, q_i^1),\ (q_i^1, \varepsilon, \varepsilon, q_i^I)\} \cup \\ & & \{(q, b, b, q') \mid \delta_i(q, b) = q'\} \cup \\ & & \{(q, \varepsilon, \varepsilon, q_i^2) \mid q \in F_i\} \cup \\ & & \{(q_i^2,),), q_i^3)\}.\end{array}$$

Finally, $\mathcal{A}_{k-1}$ is defined just as $\mathcal{A}_i$ for $0 < i < k-2$ except that the transition $(q_i^2,),), q_i^3)$ is replaced by $(q_i^2,),), {q'}_i^2)$ and $({q'}_i^2, \varepsilon, \mathsf{secret}, q_i^3)$, where ${q'}_i^2$ is a new state. Now, it easy to see that there exists a word $w \in \Sigma^*$ accepted by $\mathcal{B}_i$ for all $i < k$ iff the instance $(\mathcal{K}, \mathcal{A}_0, \ldots, \mathcal{A}_{k-1})$ of the path problem has a solution. This proves Theorem 35.

We can basically employ the same argument for ATTACK. We simply conjoin the message transducers $\mathcal{A}_i$ into one extended message transducer $\mathcal{A} = (Q, \Sigma_{\mathcal{N}}, \Delta, (I_0, \ldots, I_k))$, where Q and Δ are the union of the states and transitions of the $\mathcal{A}_i$'s, respectively, and $I_i := \{q_i^0\}$ for $i < k$, and $I_k := \{q_{k-1}^3\}$. We assume that the set of states of the $\mathcal{A}_i$'s are disjoint, except that we identify q_{i-1}^3 and q_i^0 for $0 < i < k$. Again, it is easy to see that there exists a word $w \in \Sigma^*$ accepted by $\mathcal{B}_i$ for all $i < k$ iff the transducer-based protocol with one principal defined by $\mathcal{A}$ allows a successful attack. Thus, Theorem 34 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 word w, to A_0, and then sends the message $\mathtt{enc}_a(w)$ 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 $\mathtt{enc}_a(aw)$ instead of $\mathtt{enc}_a(w)$, $\mathcal{A}_1$ only accepts a message if it is of this form and outputs $\mathtt{enc}_a(aaw)$, and so forth.

8 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 two instances, receive-send actions are modeled by sets of rewrite rules. We have shown that in these instances, 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.

While in this paper we have concentrated on the shared key setting and secrecy properties, we conjecture that our results 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 Cathrine 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), 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://citeseer.nj.nec.com/ferguson00cryptographic.html.

[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] 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.

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

[16] 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.

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

[18] L.C. Pauslon. Mechanized Proofs for a Recursive Authentication 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)*, 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, 1998.

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

A The Recursive Authentication Protocol

The recursive authentication protocol (RA protocol) was proposed by Bull and Otway [5] and it extends the authentication protocol by Otway and Rees [17] 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 [18].

Informally speaking, the protocol works as follows: We assume that a authentication server S shares long-term keys with the principals. If principal A wants to establish a session key with principal B, she sends a request message to B. In the Otway-Rees protocol, B would now contact the server S, which in turn would generate a session key and distribute it to A and B. In the recursive authentication protocol, B can also contact another principal, say C, to request a session key with C, then C can contact yet another principal, and so on, until a principal contacts S. During each round a principal adds his name and a fresh nonce to the ever-growing request message. When S obtains this message, S generates fresh keys for the pairs of principals requesting session keys, i.e., S generates a key for A and B, one for B and C, and so forth. Finally, S distributes the keys to the principals, encrypted with the respective long-term keys. What is important to note is that the request message is an open-ended data structure. It has an unbounded number of data fields, namely an unbounded sequence of pairs of principals requesting session keys, which the server needs to process in one receive-send action. Also note that the number of receive-send actions performed and principals involved in one protocol run is unbounded. For reasons pointed out above, in our model of the RA protocol we assume a fixed bound (see Appendix B).

In what follows, we give a more formal description of the RA protocol. We start with some notation.

Let $\mathtt{hash}(m)$ be the hash of m, and $\mathtt{hash}_k(m)$ the message $\mathtt{hash}(km)m$, where km is the message obtained from k and m by concatenation, and $\mathtt{hash}(km)m$ is the concatenation of $\mathtt{hash}(km)$ and m. In the protocol, k will be a long-term key shared between the server S and a principal. It is used by the server to identify the principals. In other words, $\mathtt{hash}_k(m)$ contains the message m plus the message authentication code for m computed using k.

Figure 1 depicts the protocol run informally described above: First, A contacts B, then B contacts C, and C contacts the server S. Then, the session keys generated by the server are distributed among the principals by sending messages to the principals in reverse order, i.e., S first sends a message (containing all the keys) to C, C extracts his key, and sends the remaining message to B, who does the same, and sends the remaining message to A. More precisely, the messages exchanged are of the following form.

Principal A first sends a message to B:

1. $A \to B : \mathtt{hash}_{K_a}(ABN_a-)$,

where K_a is a long-term key shared between A and S, N_a is a fresh nonce generated by A, and "$-$" indicates that this message started the protocol run. In this message A indicates that she requests a session key from the server for secure communication with B. Now, B sends something similar to C but with A's message instead of "$-$", indicating that he wants to share a session key with C.

2. $B \to C : \mathtt{hash}_{K_b}(BCN_b\mathtt{hash}_{K_a}(ABN_a-))$.

This step can 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. $C \to S : \mathtt{hash}_{K_c}(CSN_c\mathtt{hash}_{K_b}(BCN_b\mathtt{hash}_{K_a}(ABN_a-)))$.

This message is now processed by S. The outer two hashes indicate that C has called S and was called by B. Therefore, the server generates fresh keys K_{cs} and K_{bc}, intended to be used as a session key between C and S, and between B and C, respectively. Then S prepares certificates $\mathtt{enc}_{K_c}(K_{cs}SN_c)$ and $\mathtt{enc}_{K_c}(K_{bc}BN_c)$, which together with the certificates prepared later will be sent to C. Note that the key 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.

Having dealt with C's request the server discards the outer level and proceeds with the message $\mathtt{hash}_{K_b}(BCN_b\mathtt{hash}_{K_a}(ABN_a-))$. It says that B has called C and was called by A, so the server prepares two certificates $\mathtt{enc}_{K_b}(K_{bc}CN_b)$ and $\mathtt{enc}_{K_b}(K_{ab}AN_b)$. Note that K_{bc} is the same key sent to C, and K_{ab} is a fresh key generated by the server.

It remains to process the message $\mathtt{hash}_{K_a}(ABN_a-)$. It indicates that A requests a session key for communication with B, and because it contains "$-$", A must have initiated the protocol run. Thus, the server prepares only one certificate: $\mathtt{enc}_{K_a}(K_{ab}BN_a)$, where K_{ab} is the same key as the one sent to B.

Having prepared all necessary certificates, the server sends all of them to C. (The line break in the following message is only for layout purposes and does not have any meaning in the protocol.)

4. $S \to C : \mathtt{enc}_{K_c}(K_{cs}SN_c)\mathtt{enc}_{K_c}(K_{bc}BN_c)\mathtt{enc}_{K_b}(K_{bc}CN_b)\mathtt{enc}_{K_b}(K_{ab}AN_b)$
$\mathtt{enc}_{K_a}(K_{ab}BN_a)$

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

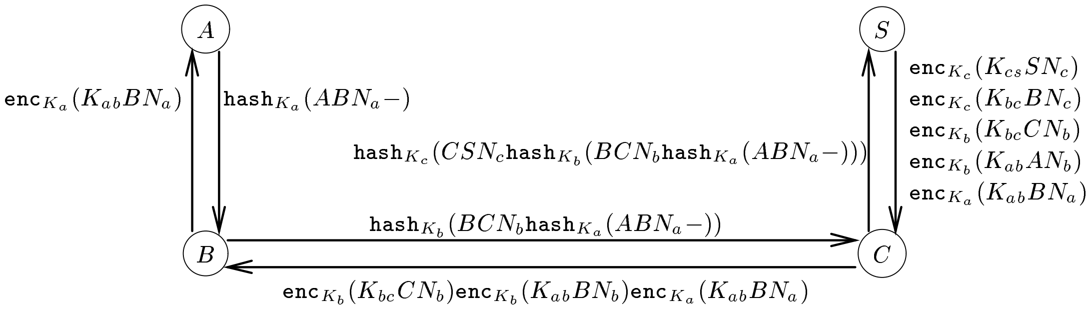

Figure 1: The Recursive Authentication Protocol

5. $C \rightarrow B$:$\texttt{enc}_{K_b}(K_{bc}CN_b)\texttt{enc}_{K_b}(K_{ab}AN_b)\texttt{enc}_{K_a}(K_{ab}BN_a)$

6. $B \rightarrow A$:$\texttt{enc}_{K_a}(K_{ab}BN_a)$

Note that in steps 4 to 6, a principal just extracts his part of the message and simply forwards the rest. For the analysis of the protocol, one can therefore assume that every principal only reads his own certificates – the intruder can do the forwarding instead. This is how the protocol will be modeled in Appendix B.

B Modeling the Recursive Authentication Protocol

We now provide a formal description of the recursive authentication protocol in the transducer-based model. To this end, we first need to simplify the messages exchanged between the principals (Appendix B.1). We then present the message transducers for the agents (Appendix B.2), i.e., all principals except the server, and in Appendix B.3 for the server.

In what follows, let $P_0, \ldots, P_n$ be the principals participating in the recursive authentication 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, $i < n$.

B.1 Simplified Messages

In order to use message transducers to model the principals, we simplify the messages exchanged. In the original protocol, as described in Appendix A, request messages have the form

$$\texttt{hash}_{K_{i_{l-1}}}(x_{l-1}\texttt{hash}_{K_{i_{l-2}}}(x_{l-2}\cdots\texttt{hash}_{K_{i_0}}(x_0)\cdots)),$$

where $\texttt{hash}_K(m) := \texttt{hash}(Km)m$. For $l = 2$, this yields

$$\texttt{hash}(K_{i_1}x_1\texttt{hash}(K_{i_0}x_0)x_0)x_1\texttt{hash}(K_{i_0}x_0)x_0.$$

The server S would check whether the hashes are taken over the correct messages, that is, the first hash in the message is really taken over K_{i_1} plus the plain text

$x_1\mathtt{hash}(K_{i_0}x_0)x_0$, and in this plain text the hash is really computed from the message $K_{i_0}x_0$. For growing l, the hashes are taken over messages of growing size. Thus, the server needs unbounded memory to check whether the hash is correct: It would first read the message inside the hash, and then compare it to the plain text message. However, a transducer has only finite memory, thus cannot perform this task. We have discussed this problem in Section 5.1. To deal with it, we can proceed in two directions.

1. We fix l, and a transducer only accepts messages with the nesting depth of the hashes restricted by l.

2. A principal simply assumes, without checking, that plain text and hash match.

The first alternative means that the recursive authentication protocol, which is an open-ended protocol, is "approximated" by a closed-ended protocol. To this approximation one could apply known formal methods. However, one would risk to miss some attacks, which may have been possible in the open-ended setting.

In the second approach, the restricted computational power of principals may lead to additional successful attacks. However, the absence of an attack guarantees that also with more powerful principals there is no attack. Therefore, we will follow this approach. In fact, the whole point of using transducers is to model principals accepting messages with an unbounded number of data fields.

Now, since transducers cannot check whether hash and plain text match, we discard the plain text altogether and only consider request messages of the following, simpler form:

$$\mathtt{hash}(K_{i_{l-1}}x_{l-1}\mathtt{hash}(K_{i_{l-2}}x_{l-2}\cdots\mathtt{hash}(K_{i_0}x_0)\cdots). \tag{5}$$

A principal P_i, $i < n$, given an input message of the form (5) will return a message of the form $x_l\mathtt{hash}(K_ix_l\mathtt{hash}(K_{i_{l-1}}x_{l-1}\mathtt{hash}(K_{i_{l-2}}x_{l-2}\cdots\mathtt{hash}(K_{i_0}x_0)\cdots))$, i.e., x_l is sent in plain text, because otherwise the intruder could not get hold of the request message. Note that in the recursive authentication protocol as described in AppendixA, the requests (e.g., "BCN_b") are also sent in plain text (together with the message authentication code).

In what follows, we show how the principals P_i, $i \leq n$, are modeled by extended message transducers. The set of atomic messages is $\mathcal{N} := \{P_i \mid i \leq n\} \cup \{K_i \mid i < n\} \cup \{N_i \mid i < n\} \cup \{K_{jj'} \mid j < n, j' \leq n\} \cup \{-\}$. The initial intruder knowledge $\mathcal{K}$ contains all the principal names plus the symbol "$-$" and the empty word ε. One could also add keys K_i and nonces N_i in case the intruder controls P_i.

B.2 The Extended message transducer of the Agents

We now define the extended message transducer $\mathcal{A}_i$ for P_i, $i < n$. The states of the transducer consist of three components. The first takes the values $\mathsf{request}$, copy, key, and accept, indicating which step is performed: $\mathsf{request}$ means that P_i sends his request message. If the message consists of nested hashes, it is necessary to copy the received message. This is done in state copy. In state key, P_i waits for the response message and extracts the session keys. Then P_i proceeds to state accept. In the second component P_i stores the name of the principal who wants to share a session key with P_i. Analogously, the third component takes the name of the principal P_i has called. The latter two components have value $\perp$, if the necessary information is not available yet.

In what follows, to increase readability, a transition (p, v, w, q) is written in the following form:

$$p \xrightarrow[w]{v} q$$

The transducer $\mathcal{A}_i$ contains the following transitions, which are labeled with words (instead of only single letters or ε as required for message transducers) in order to simplify the presentation:[8]

1. P_i initiates a protocol run and calls $P_{j'}$: For every $j' \leq n$,

$$(\mathsf{request}, \perp, \perp) \xrightarrow[P_i P_{j'} N_i \mathtt{hash}(K_i P_i P_{j'} N_i -)]{\varepsilon} (\mathsf{key}, \perp, P_{j'});$$

2. P_i is called by P_j and calls $P_{j'}$: For every $j < n$, $j' \leq n$, $a_0, a_1 \in \mathcal{N}$,

$$(\mathsf{request}, \perp, \perp) \xrightarrow[P_i P_{j'} N_i \mathtt{hash}(K_i P_i P_{j'} N_i \mathtt{hash}(a_0 P_j P_i a_1]{\mathtt{hash}(a_0 P_j P_i a_1} (\mathsf{copy}, P_j, P_{j'});$$

3. P_i copies the rest of the input message: For every $j < n$, $j' \leq n$, $a \in \Sigma_{\mathcal{N}}$,

$$(\mathsf{copy}, P_j, P_{j'}) \xrightarrow[a]{a} (\mathsf{copy}, P_j, P_{j'});$$

4. P_i concludes his request message with ")": For every $j < n$, $j' \leq n$,

$$(\mathsf{copy}, P_j, P_{j'}) \xrightarrow[)]{\varepsilon} (\mathsf{key}, P_j, P_{j'});$$

5. P_i, who initiated the protocol run, expects one certificate containing the session key for communication with P_j. The output message $\mathsf{enc}_a(\mathsf{secret})$ is used to check whether the intruder can get hold of a: For every $j' \leq n$ and $a \in \mathcal{N}$,

$$(\mathsf{key}, \perp, P_{j'}) \xrightarrow[\mathsf{enc}_a(\mathsf{secret})]{\mathsf{enc}_{K_i}(a P_{j'} N_i)} (\mathsf{accept}, \perp, \perp);$$

6. P_i reads the two certificates containing the session keys for communication with P_j and $P_{j'}$: For every $j < n$, $j' \leq n$, and $a_0, a_1 \in \mathcal{M}$,

$$(\mathsf{key}, P_j, P_{j'}) \xrightarrow[\mathsf{enc}_{a_0}(\mathsf{secret})\mathsf{enc}_{a_1}(\mathsf{secret})]{\mathsf{enc}_{K_i}(a_0 P_{j'} N_i)\mathsf{enc}_{K_i}(a_1 P_j N_i)} (\mathsf{accept}, \perp, \perp).$$

[8] Note that words read/written in one transition are not necessarily messages, i.e., the number of parenthesis may be unbalanced.

To complete the definition of the extended message transducer, it remains to specify subsets I_0, I_1, I_2 of the state space: $I_0 := \{(\mathsf{request}, \bot, \bot)\}$, $I_1 := \{(\mathsf{key}, P_j, P_{j'}) \mid j < n, j' \leq n\}$, $I_2 := \{(\mathsf{accept}, \bot, \bot)\}$.

Since the transducer defined so far is a transducer with word-transitions, we need to turn it into one with letter transitions. For all transitions, except the ones in 2., this is done as in the proof of Lemma 7. For the transitions in 2., one first outputs $\mathtt{hash}(K_i P_i P_{j'} N_i$ (letter by letter) and then simultaneously reads and writes $\mathtt{hash}(a_0 P_j P_i a_1$ (letter by letter). Thus, the outer hash is written before the inner hashes are read/written. If in 2. first the input was read and then the output (as suggested in the proof of Lemma 7), then when $\mathcal{A}_i$ performs transition 4. the closing parenthesis for the outer hash $\mathtt{hash}(K_i P_i P_{j'} N_i$ written in 2. would be written *after* the last closing parenthesis of the input. Thus, if the input is completely written, the output up to this point would not be a message. But this would violate the second property for message transducers. Nevertheless, if in 2. the transitions are translated into transitions with letters as explained above, one can easily check that the conditions for message transducers (cf. Definition 8) are satisfied.

Finally, we remark that in the model for P_i we have assumed that a principal can check whether a message is atomic. Thus, we have a restricted kind of typing. For example, in transition 2. the principal P_i only accepts the outer hash if a_0 and a_1 are atomic messages. If keys and nonces could be arbitrary messages, the transducer could not parse the nested hashes correctly.

B.3 The Extended Message Transducer of the Server

We define the extended message transducer $\mathcal{A}_n$ for the server $P_n = S$. The states consist of three components. The first takes the values start, read, $\mathsf{readpar}$, and accept. In the state start, $\mathcal{A}_n$ reads the first symbols of the message, checks whether this message is really addressed to S, and generates the first certificates. In state read, $\mathcal{A}_n$ processes the rest of the requests. At the end, $\mathcal{A}_n$ needs to read remaining closing parentheses. This is done in state $\mathsf{readpar}$. If everything is ok, S goes into the state accept. In the second component, $\mathcal{A}_n$ memorizes whose certificates are to be generated, and the third component stores the corresponding nonce.

The transitions in $\mathcal{A}_n$, again labeled with words, are specified as follows:

1. S reads the first request and generates the corresponding certificate: For every $a \in \mathcal{N}$,
$$(\mathsf{start}, \bot, \bot) \xrightarrow[\mathtt{enc}_{K_i}(K_{in} S a)]{\mathtt{hash}(K_i P_i S a} (\mathsf{read}, P_i, a);$$
2. P_i has initiated the protocol run, and therefore, no additional certificate needs to be generated: For every $i < n$ and $a \in \mathcal{N}$,
$$(\mathsf{read}, P_j, a) \xrightarrow[\varepsilon]{-} (\mathsf{readpar}, \bot, \bot);$$
3. The remaining certificate for P_i is generated and a new one for $P_{i'}$: For every

$i, i' < n$ and $a_0, a_1 \in \mathcal{N}$,

$$(\mathsf{read}, P_i, a_0) \xrightarrow[\mathtt{enc}_{K_i}(K_{i'} i P_{i'} a_0)\mathtt{enc}_{K_{i'}}(K_{i'} i P_i a_1)]{\mathtt{hash}(K_{i'} P_{i'} P_i a_1} (\mathsf{read}, P_{i'}, a_1);$$

4. The remaining closing parentheses are read:

$$(\mathsf{readpar}, \bot, \bot) \xrightarrow[\varepsilon]{)} (\mathsf{readpar}, \bot, \bot);$$

5. S is done if the last closing parenthesis is read:

$$(\mathsf{readpar}, \bot, \bot) \xrightarrow[\varepsilon]{)} (\mathsf{accept}, \bot, \bot).$$

Since S only performs a single action, we only need to define two sets I_0 and I_1: $I_0 := \{(\mathsf{start}, \bot, \bot)\}$, $I_1 := \{(\mathsf{accept}, \bot, \bot)\}$. If $\mathcal{A}_n$ is turned into a transducer with letter transitions as in the proof of Lemma 7, it is easy to see that $(\mathcal{A}_n)_{I_0, I_1}$ is a message transducer. In particular, the two conditions imposed on message transducers are satisfied.

Just as in Section B.2, we assume that S can check whether a message is atomic. For instance in 1., S only accepts the outer hash if a is atomic.

Finally, note that the nonces sent by the principals to S must be stored by S because they may occur in two certificates. That is, S must copy submessages. As explained in Section 5.1, because of the finite memory of transducers, this is only possible if the submessages to be copied have bounded size. Therefore, S, as modeled here, assumes nonces to be atomic.