Bounded Memory Protocols

Max Kanovich[1], Tajana Ban Kirigin[2], Vivek Nigam[3], Andre Scedrov[4]

Abstract

It is well-known that the Dolev-Yao adversary is a powerful adversary. Besides acting as the network, intercepting, decomposing, composing and sending messages, he can remember as much information as he needs. That is, his memory is unbounded. We recently proposed a weaker Dolev-Yao like adversary, which also acts as the network, but whose memory is bounded. We showed that this Bounded Memory Dolev-Yao adversary, when given enough memory, can carry out many existing protocol anomalies. In particular, the known anomalies arise for *bounded memory protocols*, where although the total number of sessions is unbounded, there is only a bounded number of concurrent sessions and the honest participants of the protocol cannot remember an unbounded number of facts nor an unbounded number of nonces at a time. This led us to the question of whether it is possible to infer an upper-bound on the memory required by the Dolev-Yao adversary to carry out an anomaly from the memory restrictions of the bounded protocol. This paper answers this question negatively (Theorem 8).

1. Introduction

In the symbolic verification of protocol security, one considers a powerful adversary model now usually referred to as the Dolev-Yao adversary, which arose from positions taken by Needham and Schroeder [NS78] and a model presented by Dolev and Yao [DY83]. Not only can the Dolev-Yao adversary act as the network, intercepting, decomposing, composing and sending messages, but he can also remember as much information as he needs. The goal in protocol verification is to demonstrate that such a powerful adversary cannot discover some secret information, when using some protocol(s). Clearly, if it is shown that such a powerful adversary cannot discover the secret symbolically, then weaker adversaries will also not be able to discover the secret.

In [KKNS], we proposed a Bounded Memory Dolev-Yao adversary, which is similar to the Dolev-Yao adversary. He also acts as the network, intercepting, sending and composing messages, but differently from the Dolev-Yao adversary, he can remember

Email addresses: mik@dcs.qmul.ac.uk (Max Kanovich), bank@math.uniri.hr (Tajana Ban Kirigin), vivek@ci.ufpb.br (Vivek Nigam), scedrov@math.upenn.edu (Andre Scedrov)

[1]Queen Mary, University of London, UK

[2]University of Rijeka, HR

[3]Federal University of Paraíba, João Pessoa, Brazil

[4]University of Pennsylvania, Philadelphia, USA

only a bounded number of facts at a given time. So, in order for him to learn some new information, such as a nonce, he might have to forget some information he previously learned, such as an old nonce. Clearly, our Bounded Memory Dolev-Yao adversary is weaker than the Dolev-Yao adversary, as the former's memory is bounded, while the latter's is not.

However, despite being weaker, we demonstrated in our previous work [KKNS] that many known anomalies can also be carried out by our Bounded Memory Dolev-Yao adversary. We also noticed that the protocols for which we could replay the anomaly with our bounded memory adversary were all *bounded memory protocols*, where one considers that the memory of the system is bounded. That is, in concurrent runs the honest participants of the protocol also cannot remember an unbounded number of facts nor an unbounded number of nonces at a time. This led us to the question of whether it is possible to infer an upper bound on the memory of the Dolev-Yao adversary with respect to the memory restrictions of bounded memory protocols, that is, with respect to the memory used by the participants. Such an upper bound would mean that an attack using the Standard Dolev-Yao adversary exists if and only if an attack using the Bounded Memory adversary exists.

This paper answers this question negatively. That is, it is *not possible* to determine an upper bound on the memory of the Dolev-Yao adversary even if the memory of the protocol is bounded. From our main result (Theorem 8), we can infer that the Standard Dolev-Yao adversary cannot be constructively approximated by an infinite sequence of increasing memory Bounded Memory adversaries. We show this negative result by proposing a novel undecidability proof for the secrecy problem with the Dolev-Yao adversary. Our undecidability result strengthens the one given in [CDL+99, DLMS04], confirming the hardness of protocol verification. In particular, we show that the secrecy problem is "very undecidable:" the secrecy problem is undecidable *even for bounded memory protocols* and thus a bound on the memory of the Dolev-Yao adversary is not computable from a bound on the memory used by a protocol. This is accomplished by a novel encoding of Turing machines by means of memory bounded protocols.

After Section 2 where we introduce the preliminary concepts used in this paper, including Balanced Actions, we proceed as follows:

- Section 3 reviews the specification of bounded memory protocols, the Dolev-Yao Adversary, and of Bounded Memory Adversaries. It also reviews some of the complexity results for the secrecy problem;
- Section 4 contains the secrecy undecidability proof with memory bounded protocols. This is a novel, stronger undecidability proof, which allows us to infer that it is not possible to determine an upper bound on the memory of the Dolev-Yao adversary from the memory bound of the protocol;
- Section 5 revisits the undecidability proof given in [CDL+99, DLMS04] and shows that a similar proof can be obtained by using Bounded Memory Protocols;
- Finally in Sections 6 and 7 we comment on related work and conclude by pointing out to future work.

This is an extended and improved material on bounded memory protocols from the conference paper [KKNS13]. Besides containing most of the proofs and more detailed explanation, the material in Section 5 is novel.

2. Preliminary: Configurations, Actions and Balanced Actions

We formalize bounded memory protocol theories and adversary theories by means of multiset rewrite rules, similarly as in [CDL+99, DLMS04]. A set of rewrite rules, or a theory, was proposed in [CDL+99, DLMS04] for modeling protocols and the standard Dolev-Yao adversary with unbounded memory. In order to carefully compare our complexity results, we closely follow this approach and adapt the theories from [CDL+99, DLMS04] to formalize bounded memory protocols and Bounded Memory Adversaries.

Assume fixed a sorted first-order alphabet consisting of constant symbols, $c_1, c_2, \ldots$, function symbols, $f_1, f_2, \ldots$, and predicate symbols, $P_1, P_2, \ldots$ all with specific sorts (or types). The multi-sorted terms over the signature are expressions formed by applying functions to arguments of the correct sort. A *fact* is a ground, atomic formula over multi-sorted terms. Facts have the form $P(t_1, \ldots, t_n)$ where P is an n-ary predicate symbol, where $t_1, \ldots, t_n$ are terms, each with its own sort.

The *size of a fact* is the total number of term and predicate symbols it contains. We count one for each predicate, function, constant, and variable symbols. We use $|F|$ to denote the size of a fact F. For example, $|P(x, c)| = 3$, and $|P(f(z, x, n), z)| = 6$. We will assume in the remainder of this paper an upper bound on the size of facts, as in [CDL+99, DLMS04, KRS11]. As we argue later in this section, the combination of a bound on the size of facts and the use of balanced actions imposes a bound on the memory of the system. This will be key for the decidability of the problems that we deal with in this paper.

A *state* or *configuration* of the system is a finite multiset of grounded facts, *i.e.*, facts that do not contain variables. Intuitively configurations specify the state of the world. They are modified by actions, which are in general multiset rewrite rules of the following form:

$$X_1, \ldots, X_n \longrightarrow \exists \vec{x}. Y_1, \ldots, Y_m \tag{1}$$

where the X_is and Y_js are facts. The collection $X_1, \ldots, X_n$ is called the *pre-condition* of the rule, while $Y_1, \ldots, Y_m$ is called its *post-condition*. We assume that all free variables are universally quantified at the head of the rule. By applying the rule for a ground substitution (σ), the pre-condition ($X_1\sigma, \ldots, X_n\sigma$) to which this substitution has been applied is replaced with the post-condition ($Y_1\sigma, \ldots, Y_m\sigma$) to which the same substitution has been applied. In this process, the existentially quantified variables ($\vec{x}$) appearing in the post-condition are replaced by fresh constants, also called *nonces* in protocol security literature. The rest of the configuration remains untouched. Thus, we can apply the rule $P(x), Q(y) \rightarrow \exists z. R(x, z), Q(y)$ to the global configuration $V, P(t), Q(s)$ to get the global configuration $V, R(t, c), Q(s)$, where the constant c is new.

Given a multiset rewrite system R, one is often interested in the *reachability problem*: Is there a sequence of (0 or more) rules from R which transforms configuration

W into Z? If this is the case then we say that Z is reachable from W using R, and we call any such sequence of actions a *plan*.

2.1. Balanced Actions, Empty Facts and Nonce Updates

An important condition for formalizing bounded protocols is that of *balanced actions*, introduced in the context of collaborative systems [KRS11]. We classify an action as *balanced* if the number of facts in its pre-condition is the same as the number of facts in its post-condition. That is, $n = m$ in Equation 1. If we restrict all actions in a system to be balanced, then the number of facts in each configuration in a run is the same as in the initial configuration. This is because when a balanced action is applied to a configuration, the same number of facts that are removed from the configuration are also inserted into it.

The main motivation of using balanced actions is to bound the storage capacity of agents. Since the number of facts of all configurations in a plan is the same as the number of facts in the initial configuration, denoted by the symbol m, the number of facts that are known to agents at any time is bounded. However, even with the use of balanced actions, it does not mean that there is a bound on the number of symbols (or terms) present in any configuration. One also needs a *bound on the size of facts* denoted by the symbol k. Otherwise, an agent would be able to store as many symbols as he wants by using for instance a pairing rule:

$$M(t_1), M(t_2) \rightarrow M(\langle t_1, t_2 \rangle), M(t_2)$$

Notice that this action is balanced, but the number of symbols that appear in the post-condition is greater than the number of symbols that appear in the pre-condition. Without bounding the size of facts, one would be able to store an unbounded number of symbols in the term level. In this way it would be possible for the system, represented by a configuration, to contain as much information as needed. On the other hand, bounding the size of facts implies that the pairing rule would not be allowed if the size of the fact $M(\langle t_1, t_2 \rangle)$ is greater than the upper-bound. Thus, the number of symbols in a configuration is always bounded by the value mk, where m is the number of facts in the initial configuration and k the upper-bound on the size of facts.

These two conditions, namely the use of balanced actions and of an upper-bound on the size of facts, are crucial for our decidability results:

- Kanovich, Rowe and Scedrov showed [KRS09] that for *unbalanced systems* the reachability problem is undecidable even if we bound the size of facts;
- It was also shown [DLMS04] by encoding the Post correspondence problem that *if one does not bound the size of facts*, then the reachability problem is undecidable even if the system is balanced.

In contrast, our previous work [KKNS] showed that the reachability problem is PSPACE-complete for balanced system if we assume both a balanced system and an upper-bound on the size of facts.

As an illustration, consider the following unbalanced rule:

$$\rightarrow \exists n.M(n)$$

This rule specifies that the system may create a fact with a fresh nonce increasing the number of facts in a configuration. Such a rule is not balanced. To support the creation of new facts in balanced systems, we use *empty facts*, written $P(*)$. Intuitively, empty facts denote available memory slots that could be filled with some new information. Here $*$ is not a constant, but just used for illustrative purposes. By using empty facts, one can transform unbalanced systems into balanced systems simply by adding enough empty facts to the pre-condition or the post-condition of each rule with so that it becomes balanced. For instance, a balanced version of the above rule, called GEN, is shown below:

$$\text{GEN: } P(*) \rightarrow \exists n.M(n)$$

The system can then create a new nonce provided the given configuration has at least one empty fact. It is also assumed that the system may forget some information, by replacing a fact by an empty fact, as illustrated by the following rule, called DELM:

$$\text{DELM: } M(x) \rightarrow P(*)$$

This is a simple solution for bounding the storage capacity of agents. The agents in the system may need to be careful on how they use the system's memory limitation.

Finally, the above rules also illustrate two important aspects of using nonces. The first unbalanced rule specifies that one can create a nonce, while the two balanced rules, GEN and DELM, specify that agents in a balanced system do not create nonces, but rather update an existing symbol with a fresh nonce. That is, they replace an existing symbol, possibly an existing nonce, with a fresh one. Hence, differently from the unbalanced systems, agents do not remember all the nonces created. This distinction between *nonce creation* and *nonce update* will play an important role in our undecidability proof for the secrecy problem (defined formally in the next section), when compared with the proof given by Durgin *et al.* [DLMS04]. The honest participants in our Turing machine encoding (Section 4) will rely only on nonce updates, while the honest participants in the proof given by Durgin *et al.* [DLMS04] use nonce creation.

3. Bounded Memory Protocols and Adversaries

This section reviews the definition of Bounded Memory Protocols introduced in our previous work [KKNS] illustrating the difference to well-founded protocols used in [DLMS04]. We also discuss the differences between the Standard Dolev-Yao adversary and the Bounded Memory adversary in Section 3.2 and finally we review the complexity results for the secrecy problem in Section 3.3, namely the undecidability result given by Durgin *et al.* [DLMS04] in the presence of the Standard Dolev-Yao adversary and our PSPACE-completeness [KKNS] in the presence of the Bounded Memory adversary.

3.1. Bounded Memory Protocols

A bounded memory protocol, formally defined below, only contains balanced actions [KKNS]. This means that the number of facts known by the participants at a given time is bounded. Bounding the memory available for protocol sessions also intuitively

bounds the number of concurrent protocol sessions. This is because for each protocol session, one needs some free memory slots to remember, for instance, the internal states of the agents involved in the session. However, this does not mean that there may not be an unbounded number of protocol sessions in a trace. Once a protocol session is completed, the memory slots it required can be re-used to initiate a new protocol session.

This is different from the well-founded protocol theories [CDL+99, DLMS04] proposed in the literature. The rules in well-founded protocol theories are not necessarily balanced. All protocol sessions are created at the beginning of the trace before any protocol session starts executing. Therefore, in well-founded protocol theories, an unbounded number of protocol sessions can run concurrently and therefore participants are allowed to remember an unbounded number of facts.

Definition 1 (Balanced Role Theory). *A theory $\mathcal{A}$ is a* balanced role theory *if there is a finite list of predicate names called the* role states $S_0, S_1, \ldots, S_m$ *for some m, such that every rule $L \to \exists \vec{t}.R$ in $\mathcal{A}$ is balanced and there is exactly one occurrence of a state predicate in L, say S_i, and exactly one occurrence of a state predicate in R, say S_j, such that $i < j$. We call the first role state, S_0,* initial role state*, and the last role state S_m* final role state*. Only rules with final role states can have an empty fact in the post-condition.*

Defining roles in this way ensures that each application of a rule in $\mathcal{A}$ advances the state forward. Each instance of a role can only result in a finite number of steps in a trace. The request on empty facts formalizes the fact that one of the participants, either the initiator or the responder, sends the "last" protocol message. In [KKNS], one can find several examples of protocols specified as balanced role theories.

In order to allow for an unbounded number of protocol sessions in a trace, we allow protocol roles to be created at any time with the cost of consuming empty facts $P(*)$. At the same time, we allow protocol sessions that have been completed to be forgotten. Once a final role state has been reached, it can be deleted, creating new empty facts $P(*)$ in the process. These empty facts can then be used to create new protocol roles starting hence a new protocol session. Such theories are called *role regeneration theories*.

Definition 2 (Role Regeneration Theory). *If $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are balanced role theories, a* role regeneration theory *is a set of rules that either have the form*

$$Q_1(\vec{x}_1) \cdots Q_n(\vec{x}_n) P(*) \to Q_1(\vec{x}_1) \cdots Q_n(\vec{x}_n) S_0(\vec{x}) \,,$$

where $Q_1(\vec{x}_1) \ldots Q_n(\vec{x}_n)$ is a finite list of facts not involving any role states, and S_0 is the initial role state for one of theories $\mathcal{A}_1, \ldots, \mathcal{A}_k$, or have the form

$$S_m \to P(*),$$

where S_m is the final state for one of theories $\mathcal{A}_1, \ldots, \mathcal{A}_k$.

This definition is a central difference to well-founded protocol theories [CDL+99, DLMS04]: In well-founded protocol theories [CDL+99, DLMS04] one assumes that

all protocol sessions are initialized at the beginning of the trace, that is, all protocol sessions run concurrently. This means that there is no bound on the memory of the (honest) participants since they need to remember that they participate in a possibly unbounded number of protocol sessions. Under the definition above, on the other hand, this is no longer the case as the explicit use of balanced actions in role theories and role regeneration theories allows us to bound the memory of the participants, including the number of concurrent protocols in the system, without bounding the total number of sessions in a trace.

Definition 3 (Bounded Memory Protocol Theory). *A pair* $(\mathcal{P}, H)$ *is a* bounded memory protocol theory *if* H *is a finite set of facts (called* initial set*), and* $\mathcal{P} = \mathcal{R} \uplus \mathcal{A}_1 \uplus \cdots \uplus \mathcal{A}_n$ *is a protocol theory where* $\mathcal{R}$ *is a role regeneration theory involving only facts from* H *and the initial and final roles states of* $\mathcal{A}_1, \ldots, \mathcal{A}_n$*, and* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *are balanced role theories. For role theories* $\mathcal{A}_i$ *and* $\mathcal{A}_j$*, with* $i \neq j$*, no role state predicate that occurs in* $\mathcal{A}_i$ *can occur in* $\mathcal{A}_j$*.*

Intuitively, a bounded memory protocol theory specifies a particular scenario to be model-checked involving some given protocol(s). Besides empty facts, $P(*)$, the finite initial set of facts (H) contains all the facts with the information necessary to start protocol sessions, for instance, shared and private keys, the names of the participants, as well as any compromised keys. Here, for simplicity, we assume only symmetric keys, although other types of keys can be also formalized following the lines described in [CDL+99, DLMS04].

3.2. Standard Dolev-Yao and Bounded Memory Dolev-Yao Adversaries

The powerful adversary proposed by Dolev and Yao [DY83] acts as the network, where all messages communicated are sent through the adversary. He hears everything and learns messages modulo encryption. More precisely, he is capable of intercepting any message sent by a protocol participant and can then store the received information, decompose it and decrypt with the keys he possesses. He cannot, however, decrypt messages for which he does not have the correct key. Moreover, he can also create fresh values, encrypt, compose messages from the information he has learned and send messages. One of his major strengths is that he can remember as much information as he wants, *i.e.*, *his memory is unbounded*.

Figure 1a. depicts the rules of such an adversary. The I/O rules specify the fact that the adversary acts as the network receiving all messages sent (N_S) and sending all messages that are to be received (N_R). The remaining rules are straightforward, specifying when the adversary may decompose and compose messages. The term

$$E_K(t)$$

denotes the message obtained by encrypting the term t with the key K. Notice that contrary to the formalization of the bounded memory protocols, the actions specifying the Dolev-Yao adversary are not all balanced. In particular, the adversary may always learn new facts, such as in the actions DECS and GEN, where the adversary learns the contents of an encrypted message and creates a nonce respectively.

I/O Rules:

REC : $N_S(x) \rightarrow M(x)$

SND : $M(x) \rightarrow N_R(x)$

Decomposition Rules:

DCMP : $M(\langle x, y\rangle) \rightarrow M(x)\ M(y)$

DECS : $M(k)\ M(E_k(x)) \rightarrow M(k)\ M(E_k(x))\ M(x)$

Composition Rules:

COMP : $M(x)\ M(y) \rightarrow M(\langle x, y\rangle)$

USE : $M(x) \rightarrow M(x)\ M(x)$

ENCS : $M(k)\ M(x) \rightarrow M(k)\ M(E_k(x))$

GEN : $\rightarrow \exists n.M(n)$

(a) Theory for the Standard Dolev-Yao Adversary

I/O Rules:

REC: $N_S(x) \rightarrow M(x)$

SND: $M(x) \rightarrow N_R(x)$

Decomposition Rules:

DCMP: $M(\langle x, y\rangle)\ P(*) \rightarrow M(x)\ M(y)$

DEC: $M(k)\ M(E_k(x))\ P(*) \rightarrow M(k)M(x)\ M(E_k(x))$

Composition Rules:

COMP: $M(x)\ M(y) \rightarrow M(\langle x, y\rangle)\ P(*)$

USE: $M(x)P(*) \rightarrow M(x)M(x)$

ENC: $M(k)M(x) \rightarrow M(k)\ M(E_k(x))$

GEN: $P(*) \rightarrow \exists n.M(n)$

Memory maintenance rule:

DELM: $M(x) \rightarrow P(*)$

(b) Bounded Memory Dolev-Yao Adversary Theory

Figure 1: Theories for the Standard and the Bounded Memory Adversaries

In [KKNS], we proposed a Bounded Memory Dolev-Yao adversary, which has many capabilities of the Dolev-Yao adversary. He can intercept, send and compose messages, create nonces, etc. But differently from the Dolev-Yao adversary, he can remember only a bounded number of facts of a bounded size, at any given time. This is formally imposed by the balanced adversary theory presented in Figure 1b and by assuming a bound on the size of all facts. Such a bound disables the intruder to form terms of unbounded size and use them for storing unbounded amount of data inside facts.

In order for intruder to store some new information, such as a nonce, he might have to forget some information he previously learned. This is technically accomplished by using empty facts $P(*)$. For instance, the DCMP rule specifies that the bounded memory adversary can decompose a pair he learned ($M(\langle t_1, t_2\rangle)$) only if he has an empty fact left. So in order to carry out an anomaly, the adversary may need to manage his memory, by forgetting some data he previously learned. This is specified by the additional memory maintenance rule (DELM). Notice that by using the memory maintenance rule adversary is able to replace any M-fact, *i.e.* he may choose to forget any data from his memory. For example he can forget some message from an earlier session, parts of messages, nonces etc. On the other hand, if in some interaction with a protocol, adversary has enough free memory to perform his actions, it's not necessary for him to use these rules, *i.e.* he's not forced to forget any data.

3.3. Complexity Results for the Secrecy Problem

In an interaction of malicious adversaries with honest participants, one is interested in the *secrecy problem*, namely, in determining whether the adversary can discover a secret s. Formally it is an instance of the reachability problem: Is it the case that a

configuration containing $M(s)$ can be reached from an initial configuration, where s is a secret originally owned by an honest participant?

Undecidability of the Secrecy Problem. It has been known for some time that the secrecy problem is undecidable in general [CDL+99, DLMS04]. The undecidability proof in [CDL+99, DLMS04] proceeds by encoding the existential Horn problem, which is also proved to be undecidable. However, that encoding used *well-founded protocol theories*, where the memory of the protocol was unbounded. For instance, in well-founded protocol theories, it is allowed for an unbounded number of concurrent protocol sessions to run at the same time. In fact, all the protocol sessions in a trace are initialized at the beginning before any session starts. This implies that the participants of the system may remember an unbounded number of facts, namely, the facts containing the information of the protocol sessions in which they are participating.

In Section 4, we strengthen the result in [CDL+99, DLMS04], by showing that the secrecy problem is undecidable *even if the memory of the protocol is bounded.* This is accomplished by a novel encoding of Turing machines by means of memory bounded protocols. In Section 5, we revisit the undecidability proof given in [DLMS04] and show that it is also possible to encode the existential Horn problem by using bounded protocol theories.

4. Protocol security is very undecidable: A bound on the adversary cannot be inferred from a bound on a protocol

We now detail the sound and faithful encoding of Turing machines using bounded memory protocols. We show that an attack on the given protocol by *an unbounded, standard Dolev-Yao adversary* is possible if and only if the encoded Turing machine terminates. From that we infer the undecidability of a Dolev-Yao attack *even for bounded memory protocols*. Notice that our result works even if we assume a (large enough) bound on the size of facts, *e.g.*, a bound around 30.

4.1. Encoding of Turing Machine Tapes

Without loss of generality, let $\mathcal{M}$ be a Turing machine such that

(i) $\mathcal{M}$ has only one tape, which is one-way unbounded to the right. The leftmost cell (numbered by 0) contains the marker \$ unerased;

(ii) The initial 3-cell configuration is of the following form, where B stands for the blank symbol:

\$	$\langle q_1, B\rangle$	B

(2)

We write $\boxed{\langle q, \xi\rangle}$ to denote that the corresponding cell contains the symbol ξ and is scanned by $\mathcal{M}$ in its state q.

(iii) We assume that all instructions of $\mathcal{M}$ are "move" instructions, *i.e.* of the form $q\xi \to q'\eta R$ or $q\xi \to q'\eta L$, denoting: *"if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the right, respectively to the left, and go into state q'"*.

(iv) The head of $\mathcal{M}$ cannot move to the leftmost cell marked with \$.

(v) Finally, $\mathcal{M}$ has only one *accepting* state, q_0.

Encoding of the Tape. In our encoding, we need two honest participants only, Alice and Bob. Assume they share a symmetric key K, not known to any other participant and that only Alice knows a secret s. We will encode the tape cells separately as follows:

(a) An unscanned cell that contains symbol ξ_0 is encoded by a term encrypted with the key K

$$E_K(\langle t_0, \xi_0, e_0, t_1\rangle),$$

where t_0 and t_1 are nonces, and $e_0 = 1$ if the cell is the last cell in a configuration and $e_0 = 0$ otherwise.

(b) The cell that contains symbol ξ and is scanned by $\mathcal{M}$ in state q is also encoded by a term encrypted with the key K

$$E_K(\langle t_1, \langle q, \xi\rangle, 0, t_2\rangle)$$

where t_1 and t_2 are nonces. In the above encoding of the scanned cell, the symbol 1 is never used in the place of symbol 0, since the head never visits the last cell of the tape. As will be shown below, as soon as the head visits the penultimate cell the tape is extended.

(c) Adjacent cells share nonces as follows. The encoding of the cell immediately to the right of the cell encoded by $E_K(\langle t, \alpha, 0, t'\rangle)$ is $E_K(\langle t', \beta, e_0, t''\rangle)$.

Motivation: The nonces t_0 and t_1 in the terms $E_K(\langle t_0, \alpha, e_0, t_1\rangle)$ encoding the tape cells are used for two purposes:

(i) Firstly, t_0 and t_1 serve as "timestamps" of a visit made by $\mathcal{M}$ in the cell. Whenever $\mathcal{M}$ re-visits this cell, the previous term is updated with fresh nonces indicating a new visit;

(ii) Secondly, as t_0 and t_1 are unique, they are used to uniquely link cells that are adjacent to each other.

For example, the initial configuration, Equation (2), with three cells is encoded by using the sequence of nonces t_0, t_1, t_2, t_3 as shown below:

$$\langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q_1, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle)\rangle$$

Notice the role of the nonces t_0, t_1, t_2, t_3. In particular, the nonces t_1 and t_2 are used to correctly encode the fact that the cell $\langle q_1, B\rangle$ is to the right of the cell with the mark \$ and to the left of the cell with the blank symbol.

Initial set of facts:

$$P(*),\ P(*),\ P(*),\ Guy(A,K),\ Guy(B,K),\ Secret(s),\ Failure(f),$$
$$Init(\langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle)$$

Role Theory for Alice:

UPDA: $A_0(X,k,s,f)P(*) \rightarrow \exists \vec{t_i'}.A_1(k,s,f)N_S(X')$
RESA0: $A_1(k,s,f)N_R(Y_0) \rightarrow A_2(Y_0,k,s,f)N_S(s)$
RESA1: $A_1(k,s,f)N_R(Y_1) \rightarrow A_2(Y_1,k,s,f)N_S(s)$
RESA2: $A_1(k,s,f)N_R(Y_2) \rightarrow A_2(Y_2k,s,f)N_S(s)$
RESA3: $A_1(k,s,f)N_R(Y_3) \rightarrow A_2(Y_3,k,s,f)N_S(f)$

where

$$\begin{array}{lcl}
X & = & \langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
X' & = & \langle E_K(\langle t_0', \$, 0, t_1'\rangle), E_K(\langle t_1', \langle q, B\rangle, 0, t_2'\rangle), E_K(\langle t_2', B, 1, t_3'\rangle), E_K(\langle t_4', B, 1, t_5'\rangle)\rangle \\
Y_0 & = & \langle E_K(\langle t_{00}, \langle q_0, \xi\rangle, 0, t_{01}\rangle), E_K(\langle t_{01}, \alpha_1, 0, t_{20}\rangle), E_K(\langle t_{20}, \alpha_2, e_2, t_{21}\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
Y_1 & = & \langle E_K(\langle t_{00}, \alpha_0, 0, t_{01}\rangle), E_K(\langle t_{01}, \langle q_0, \xi\rangle, 0, t_{20}\rangle), E_K(\langle t_{20}, \alpha_2, e_2, t_{21}\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
Y_2 & = & \langle E_K(\langle t_{00}, \alpha_0, 0, t_{01}\rangle), E_K(\langle t_{01}, \alpha_1, 0, t_{20}\rangle), E_K(\langle t_{20}, \langle q_0, \xi\rangle, e_2, t_{21}\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
Y_3 & = & \langle E_K(\langle t_{00}, \alpha_0, 0, t_{01}\rangle), E_K(\langle t_{10}, \alpha_1, 0, t_{12}\rangle), E_K(\langle t_{20}, \alpha_2, e_2, t_{21}\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
 & & \text{where } \alpha_i \neq \langle q_0, \xi\rangle,\ i = 0,1,2
\end{array}$$

Role Theory for Bob:

ROLB: $Guy(G,k)P(*) \rightarrow Guy(G,k)B_0(k)$
UPDB: $B_0(k)N_R(X) \rightarrow \exists \vec{t_i'}.B_1(k)N_S(X')$
MOVEB$_\varrho$: $B_0(k)N_R(Y) \rightarrow \exists \vec{t_i'}B_1(k)N_S(Y')$ for each of $\mathcal{M}$'s instruction ϱ

where

$$\begin{array}{lcl}
X & = & \langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle t_1, \langle q, \xi\rangle, 0, t_2\rangle), E_K(\langle t_2, \xi_2, 1, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
X' & = & \langle E_K(\langle t_0', \xi_0, 0, t_1'\rangle), E_K(\langle t_1', \langle q, \xi\rangle, 0, t_2'\rangle), E_K(\langle t_2', \xi_2, 0, t_3'\rangle), E_K(\langle t_3', B, 1, t_4'\rangle)\rangle \\
Y & = & \langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle t_1, \langle q, \xi\rangle, 0, t_2\rangle), E_K(\langle t_2, \xi_2, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
Y' & = & \langle E_K(\langle t_0, \xi_0, 0, t_1'\rangle), E_K(\langle t_1', \eta, 0, t_2'\rangle), E_K(\langle t_2', \langle q', \xi_2\rangle, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
 & & \text{if } \mathcal{M}\text{'s instruction } \varrho \text{ is of the form } q\xi \rightarrow q'\eta R \\
Y' & = & \langle E_K(\langle t_0, \langle q', \xi_0\rangle, 0, t_1'\rangle), E_K(\langle t_1', \eta, 0, t_2'\rangle), E_K(\langle t_2', \eta_2, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle \\
 & & \text{if } \mathcal{M}\text{'s instruction } \varrho \text{ is of the form } q\xi \rightarrow q'\eta L
\end{array}$$

Role Regeneration Theory:

ROLA: $Guy(G,k)Init(I)Secret(s)Failure(f)P(*)$
$\rightarrow Guy(G,k)Init(I)Secret(s)Failure(f)A_0(I,k,s,f)$
ROLB: $Guy(G,k)P(*) \rightarrow Guy(G,k)B_0(k)$
ERASEA: $A_2(Y,k,s,f) \rightarrow P(*)$
ERASEB: $B_1(k) \rightarrow P(*)$

Figure 2: Bounded Memory Protocol Theory encoding the Turing Machine $\mathcal{M}$.

4.2. Encoding Turing Machine's Actions as a Bounded Memory Protocol

Given a Turing machine $\mathcal{M}$ and the encoding of the tape discussed above, we encode its actions by means of bounded memory protocol called $\mathcal{P}_{\mathcal{M}}$, which is depicted in detail in Figure 2.

Alice starts the computation by sending the message encoding the initial 3-cell configuration of $\mathcal{M}$ to Bob. Bob simulates the computation of $\mathcal{M}$ by transforming the message encoding that *in state q the symbol ξ is being read* into the new message encoding that *the state changed to q', ξ was replaced by η, and that the head moved to the right (resp. left)*, provided that $q\xi \to q'\eta R$ (resp. $q\xi \to q'\eta L$) is an instruction of $\mathcal{M}$. In one session Bob simulates one instruction of the machine, or, when necessary, models extending of the tape. Alice also checks whether the accepting state has been reached. In the modeling of the Turing machine computation, intruder plays an essential part by intercepting, storing and sending messages encoding the machine configuration. While the memory of the system used by the protocol $\mathcal{P}_{\mathcal{M}}$ is bounded, intruder's memory is unbounded enabling him to store the entire Turing machine tape and computation.

We now describe the role of Alice (initiator) and Bob (responder) in detail. Both Alice and Bob can input and output only messages of the form

$$\langle E_K(\langle t_0, \alpha_0, 0, t_1\rangle), E_K(\langle t_1, \alpha_1, 0, t_2\rangle), E_K(\langle t_2, \alpha_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

where the first three components represent the chain of three cells, and the fourth component is an auxiliary component that serves for extending the tape. This will be modeled through Bob. Namely, when Bob receives the message encoding the last three cells of the tape, he replies with the encoding denoting a new blank cell being added to the right.

Alice's Role. Assume that Alice is the initiator and her initial state contains the encoding of the initial 3-cell configuration:

$$I = \langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Because of the initial configuration of the machine (2) the first component represents the beginning of the tape and the third component represents the last cell, denoted by the symbol 1. Notice that the last term does not share nonces with the first three. It will be used for extending the tape.

The protocol starts by Alice updating all nonces t_i to t'_i, and sending the following message to Bob, action UPDA in Figure 2:

$$X' = \langle E_K(\langle t'_0, \$, 0, t'_1\rangle), E_K(\langle t'_1, \langle q, B\rangle, 0, t'_2\rangle), E_K(\langle t'_2, B, 1, t'_3\rangle), E_K(\langle t'_4, B, 1, t'_5\rangle)\rangle$$

At this point, she does not need to remember the previous terms containing the nonces t_i. That is, she erases her memory and is ready to store new facts containing the nonces t'_i. In particular, she is waiting for a message from Bob of the form:

$$\langle E_K(\langle t_0, \alpha_0, 0, t_1\rangle), E_K(\langle \widetilde{t_1}, \alpha_1, 0, \widetilde{t_2}\rangle), E_K(\langle t_2, \alpha_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

By verifying its integrity with $(t_1 = \widetilde{t}_1)$ and $(\widetilde{t}_2 = t_2)$, Alice assumes that there is no intrusion in the channel. If some α_i is of the form $\langle q_0, \xi\rangle$ containing the final state q_0, then Alice openly sends the secret s to Bob, formalized by actions RESA0, RESA1 and RESA2 in Figure 2. Otherwise, Alice sends a neutral message, see action RESA3 in Figure 2. Notice that in Ys in Figure 2 related to these rules it is the case that $\xi \neq \$$ since, by assumption on $\mathcal{M}$, the head cannot move to the cell marking the beginning of the tape.

Notice that Alice's role is given only for the encoding of the initial 3-cell configuration. Alice only checks whether the nonces from the received message match. She doesn't compare any data received with what she had sent. During the attack, described further below, Bob's replies will be intercepted and stored by the adversary.

Bob's role. The role of Bob is to transform the message received with the help of an instruction from the given Turing machine $\mathcal{M}$. Bob is expecting to receive a message (presumably from Alice) of the form:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle \widetilde{t}_1, \langle q, \xi\rangle, 0, \widetilde{t}_2\rangle), E_K(\langle t_2, \xi_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Bob verifies its integrity by $(t_1 = \widetilde{t}_1)$ and $(\widetilde{t}_2 = t_2)$, and follows one of three cases:

(1) (Extending the tape, action UPDB in Figure 2) For $e_2 = 1$, *i.e.* for the encoding of the last three cells of the tape, Bob replaces nonces t_1 through t_5 with the new ones, and sends the following updated message to Alice

$$\langle E_K(\langle t_0, \xi_0, 0, t'_1\rangle), E_K(\langle t'_1, \langle q, \xi\rangle, 0, t'_2\rangle), E_K(\langle t'_2, \xi_2, 0, t'_3\rangle), E_K(\langle t'_3, B, 1, t'_4\rangle)\rangle$$

which provides a new last cell in the chain of four cells. Notice that the fourth component in the above reply refers to the new last cell in the configuration. In particular, notice the nonces t'_1, t'_2 and t'_3 linking the adjacent cells.

(2) (Moving the Head of the Machine to the Right, action MOVEB$_\varrho$ in Figure 2) For an instruction ϱ of $\mathcal{M}$ of the form $q\xi \to q'\eta R$, denoting: *"if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the right, and go into state q'"*, Bob replaces nonces t_1 and t_2 with fresh nonces t'_1 and t'_2 respectively, to mark the event of the head moving to the right and making a new visit to the right cell. and he sends the following updated message

$$\langle E_K(\langle t_0, \xi_0, 0, t'_1\rangle), E_K(\langle t'_1, \eta, 0, t'_2\rangle), E_K(\langle t'_2, \langle q', \xi_2\rangle, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

to Alice encoding the new 3-cell configuration.

(3) (Moving the Head of the Machine to the Left, action MOVEB$_\varrho$ in Figure 2) For an $\mathcal{M}$'s instruction ϱ of the form $q\xi \to q'\eta L$, denoting: *"if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the left, and go into state q'"*, Bob replaces nonces t_1 and t_2 with the new onces, and sends the following updated message to Alice

$$\langle E_K(\langle t_0, \langle q', \xi_0\rangle, 0, t'_1\rangle), E_K(\langle t'_1, \eta, 0, t'_2\rangle), E_K(\langle t'_2, \xi_2, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Remark 4. *The above protocol is balanced. It can be formalized by a bounded memory protocol, see Figure 2. In particular, only terms of height fixed in advance are used. Also, a fixed number of facts is used by the protocol participants. No new memory is created in the system. Freshly created values are used to only update the facts in the system configuration, that is, the old nonces are replaced by new ones.*

The protocol formalization with bounded memory theories for the participants A and B is given in Figure 2. The initial set of facts contains the facts $Guy(A, K)$, $Guy(B, K)$, denoting agents Alice and Bob, and specifying that they share the uncompromised key K, the fact $Secret(s)$ denoting the secret s, the fact $Failure(f)$ representing the neutral message f (this will be used by Alice to mark that the accepting state hasn't yet been reached). Finally, $\mathcal{M}$'s initial configuration is encoded in fact $Init$. In specifications of the role theories, for convenience we use various X and Y abreviations and ξ and η variables for tape symbols. Both theories for A and for B have the corresponding role generation rules ROLA and ROLB, which create new sessions, as well as rules ERASEA and ERASEB, which delete role state predicates of completed sessions. As previously discussed, this allows traces to have an unbounded number of protocol sessions.

The machine computation is initiated by Alice's role which relates to the initial tape configuration, and is then simulated instruction per instruction through series of Bobs roles. In each session Bob either simulates one of the machine instructions ϱ through MOVEB$_\varrho$ rule, or extends the tape when necessary using UPDB rule. Bob outputs messages encoding the updated machine configuration. These messages are intercepted by intruder who, by storing and decomposing the components, is then able to produce the message for Bob's next session, *i.e.* for the simulation of the next $\mathcal{M}$'s instruction. At the same time, each of the Bob's output messages is forwarded to Alice so that she can check whether the configuration if the accepting one, that is whether the computation ends.

Since there is a MOVEB$_\varrho$ rule for each instruction ϱ of $\mathcal{M}$, the reduction is polynomial on the number of instructions in $\mathcal{M}$.

Finally notice that the initial set of facts, besides the names of the agents and the encoding of initial tape, also contains three empty facts. One is used to create a protocol role for Alice another for Bob and finally the third for running the protocol.

4.3. A Man-in-the-Middle Attack by Mallory

We now describe how a standard Dolev-Yao adversary can carry out an attack on the protocol described above. Recall that the Dolev-Yao adversary acts as the network, that is, all the messages are sent through the adversary and that his memory is unbounded.

Notice that, beacause of the form of messaegs that Alice and Bob exchange and because of the encryption under the secret key K, by active eavesdropping Mallory can accumulate terms of the form

$$E_K(\langle t_1, \alpha_1, e_1, t_2 \rangle) \tag{3}$$

if and only if they are components of outputs generated by Alice or by Bob. We now discuss the following attack on the above protocol:

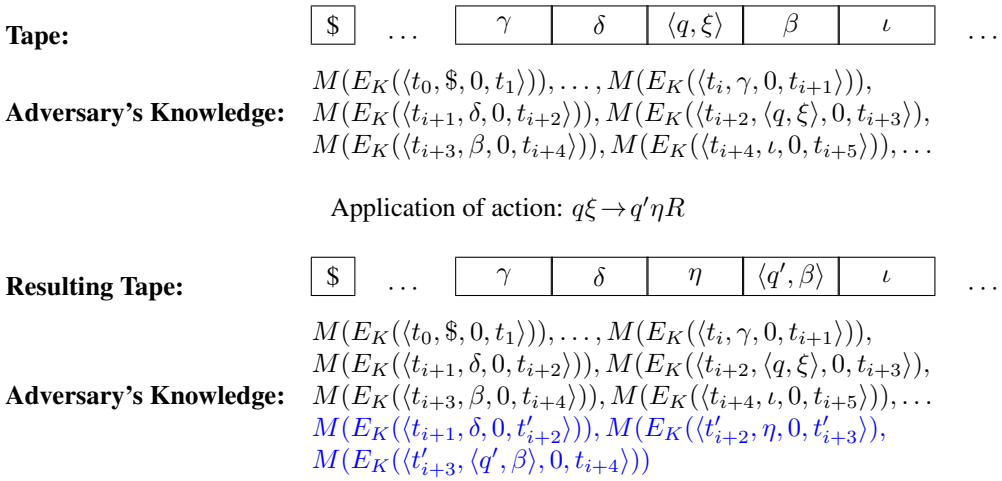


Figure 3: Illustration of the evolution of the Adversary's knowledge after a protocol session that modifies the Turing Machine Tape. The facts colored in blue are the new facts learned by the adversary.

(1) In the first session, Mallory intercepts the initial message from Alice, stores it, and resends it to Bob. While Bob responds, Mallory intercepts the message from Bob, stores it, and resends it to Alice.

(2) For each of the next sessions, Mallory first intercepts the initial message from Alice. Taking non-deterministically terms of the form (3) from his memory, Mallory then composes a message of the form:

$$\langle E_K(\langle t_0, \alpha_0, 0, t_1\rangle), E_K(\langle \widetilde{t}_1, \alpha_1, 0, \widetilde{t}_2\rangle), E_K(\langle t_2, \alpha_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

and sends it to Bob. If Bob accepts this message and responds with a transformed one as described by the protocol, then Mallory intercepts this new message from Bob, stores it, and resends it to Alice.

Recall that Alice releases the secret s after receiving the encoding of the final state of the machine $\mathcal{M}$. Since the machine configuration encoding involves fresh values and is encrypted under the secret key K, only Bob is able to update the machine configuration. Hence, Mallory will learn the secret s only if Bob outputs the encoding of the accepting state of the machine $\mathcal{M}$ *i.e.* if $\mathcal{M}$ terminates on the empty input.

Also recall that Alice's role always relates to the initial tape configuration. Bob simulates the machine computation, one instruction per session, extending the tape as necessary. Bob's replies encode the current, updated, machine configuration. Only in the first session of the anomaly Mallory forwards to Bob the encoding of the initial machine configuration. Instead, in the subsequent sessions Mallory sends to Bob the message encoding the current machine configuration. This enables the simulation of the machine computation, instruction per instruction through series of Bob's roles. In each of the sessions Mallory is able to reproduce the encoding of the updated machine configuration. Namely, if in the previous session Bob simulated a move of the machine

head, Mallory resends his last message back to Bob. Otherwise, in case in the last session Bob extended the tape, Mallory uses three E_K-terms from Bob's last message and for the fourth E_K-term Mallory can use some old term from her memory, *e.g.* a term $E_K(\langle t''_4, B, 1, t''_5\rangle)$ from one of Alice's messages. Although Alice and Bob erase their memory forgetting what they learned from exchanged messages, Mallory intercepts and stores all messages. Her memory is unbounded.

Figure 3 illustrates the knowledge that the adversary accumulates during a successful run of the protocol. Notice that although the adversary only learns the encrypted messages containing the new contents of the tape, he has all the contents of the resulting tape. This is because the nonces t_{i+1} and t_{i+4} are not modified by Bob. This intuition is formalized by the Lemma below.

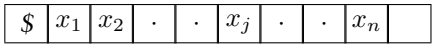
Lemma 5. *Suppose that a term of the form $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$ appears in the adversary memory by active eavesdropping. Then there is a unique sequence of nonces t_0, t_1,..., t_{n+2} and a chain of terms from the adversary's memory*

$$E_K(\langle t_0, \$, 0, t_1\rangle),\ E_K(\langle t_1, x_1, 0, t_2\rangle), \ldots . E_K(\langle t_{j-1}, x_{j-1}, 0, t_j\rangle),$$
$$E_K(\langle t_j, \langle q, x_j\rangle, 0, t_{j+1}\rangle),\ E_K(\langle t_{j+1}, x_{j+1}, 0, t_{j+2}\rangle), \ldots,\ E_K(\langle t_n, x_n, 0, t_{n+1}\rangle),$$
$$E_K(\langle t_{n+1}, B, 1, t_{n+2}\rangle)$$

such that

(a) $t_j = t,\ x_j = \xi$, *and* $t_{j+1} = t'$,

(b) $\mathcal{M}$ *leads from the empty initial configuration to the configuration where the string $x_1x_2..x_j..x_n$, is written in cells* 1, 2,..,j,..,n *on the tape*



and the j-th cell is scanned by $\mathcal{M}$ in state q.

Proof. By induction on the number of actions performed by Bob to output a message one of the components of which is $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$. Notice that any term of the form $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$ in Mallory's memory comes from an intercepted message since it is encrypted with the key K not known to Mallory. Also notice that Mallory learns this term by decomposing protocol messages. As per protocol specification this term is always accompanied by other three terms from the encoding.

For the base case, when the number of Bob's actions is 0, a term of the form $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$ must have been sent by Alice as a part of an encoding of the initial machine configuration, *i.e.* as a part of a message of the form:

$$\langle E_K(\langle t''_0, \$, 0, t''_1\rangle), E_K(\langle t''_1, \langle q, B\rangle, 0, t''_2\rangle), E_K(\langle t''_2, B, 1, t''_3\rangle), E_K(\langle t''_4, B, 1, t''_5\rangle)\rangle$$

The first three terms in this message contain a sequence of nonces $t''_0, t''_1, t''_2, t''_3$ and from the desired chain of terms in Mallory's memory corresponding to the empty computation.

Assume that Bob has performed k actions encoding machine instructions or tape extension and that there is a term of the form $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$ in Mallory's memory. If this term comes form one of Alice's messages, then the claim stands for the encoding of an empty computation, just like in the base case. In case the term is a part of a message sent by Bob in some session, but not in the last one, then the claim follows by inductive assumption. In the remaining case the term of the form $E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle)$ has been sent by Bob in the last session. Let's assume that in the last session Bob performed the action corresponding to machine head moving to the right, $q'\xi' \to q\xi R$. The remaining cases are proven similarly. In the last session Bob sent the message of the form:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle t_1, \xi_1, 0, t\rangle), E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

This was necessarily a reply to a message of the form:

$$\langle E_K(\langle t_0, \xi_0, 0, t_2\rangle), E_K(\langle t_2, \langle q', \xi'\rangle, 0, t_3\rangle), E_K(\langle t_3, \xi, e_2, t'\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

encoding the tape cells from the previous machine configuration. Such message must have been sent by Bob in the previous session. Since Mallory controls the network, she had intercepted and stored this message, and was able to decompose it so that the term $E_K(\langle t_2, \langle q', \xi'\rangle, 0, t_3\rangle)$ appears in her memory. Here we assume that Mallory is actively eavesdropping, decomposing whatever she can. Applying the induction hypothesis to $E_K(\langle t_2, \langle q', \xi'\rangle, 0, t_3\rangle)$ corresponds a sequence of nonces t'_0, t'_1,..., t'_{n+2} and the following chain of terms from Mallory's memory

$$E_K(\langle t'_0, \$, 0, t'_1\rangle),\ E_K(\langle t'_1, x'_1, 0, t'_2\rangle), \ldots . E_K(\langle t'_{j-1}, x'_{j-1}, 0, t'_j\rangle),$$
$$E_K(\langle t'_j, \langle q', x'_j\rangle, 0, t'_{j+1}\rangle),\ E_K(\langle t'_{j+1}, x'_{j+1}, 0, t'_{j+2}\rangle), \ldots,\ E_K(\langle t'_n, x'_n, 0, t'_{n+1}\rangle),$$
$$E_K(\langle t'_{n+1}, B, 1, t'_{n+2}\rangle)$$

where $t'_{j-1} = t_0,\ x'_{j'1} = \xi_0, t'_j = t_2,\ x'_j = \xi',\ t'_{j+1} = t_3,\ x'_{j+1} = \xi$, and $t'_{j+2} = t'$, such that $\mathcal{M}$ leads from the empty initial configuration to the configuration where the string $x'_1 x'_2..x'_j..x'_n$, is written in cells 1, 2,..,j,..,n on the tape and the j-th cell is scanned by $\mathcal{M}$ in state q'. Then the following sequence of nonces t'_0,..., t'_{j-1}, t_1, t, t'_{j+2}, ..., t'_{n+2} and the following chain of terms from Mallory's memory

$$E_K(\langle t'_0, \$, 0, t'_1\rangle),\ E_K(\langle t'_1, x'_1, 0, t'_2\rangle), \ldots . E_K(\langle t'_{j-2}, x'_{j-2}, 0, t_0\rangle),$$
$$E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle t_1, \xi_1, 0, t\rangle), E_K(\langle t, \langle q, \xi\rangle, 0, t'\rangle), \rangle$$
$$E_K(\langle t', x'_{j+2}, 0, t'_{j+3}\rangle), \ldots,\ E_K(\langle t'_n, x'_n, 0, t'_{n+1}\rangle), E_K(\langle t'_{n+1}, B, 1, t'_{n+2}\rangle)$$

represent the computations of $\mathcal{M}$ which leads from the empty initial configuration to the configuration where the string $x'_1 x'_2..x'_{j-1}\xi_1, \xi, x_{j+2}..x'_n$, is written in cells 1, 2,..,n on the tape and the $(j+1)$-st cell is scanned by $\mathcal{M}$ in state q. ■

Theorem 6. *There is a Dolev-Yao attack on the above protocol if and only if the machine $\mathcal{M}$ terminates on the empty input.*

Proof. We prove both directions.

(a) The direction from a terminating computation to an attack is straightforward by induction on the length of the computation. The key of the encoding is the use of nonces and the encryption under the secret key K which is not known to the adversary. Only Bob is able to update the encoding of the machine configuration. To each machine action corresponds a session of Bob's role, with additional roles for tape extension as necessary. When the machine terminates, *i.e.* reaches the final state q_0, Alice sends the secret s unencrypted. Therefore, Mallory is able to obtain the secret.

(b) The inverse direction relies on Lemma 5. Mallory will learn the secret s only if Alice receives the encoding of the accepting state of the machine $\mathcal{M}$ which must contain a term of the form $E_K(\langle \widetilde{t}_1, \langle q_0, \xi\rangle, 0, \widetilde{t}_2\rangle)$. Since adversary controls the network, in the case of a successful attack, a term of the form $E_K(\langle \widetilde{t}_1, \langle q_0, \xi\rangle, 0, \widetilde{t}_2\rangle)$, must appear in the adversary's memory. Then by Lemma 5, if a term of the form $E_K(\langle \widetilde{t}_1, \langle q_0, \xi\rangle, 0, \widetilde{t}_2\rangle)$ appears in the adversary's memory, then $\mathcal{M}$ leads from the empty initial configuration to a final configuration where a cell is scanned in state q_0. That is, machine $\mathcal{M}$ terminates on the empty input. ■

Notice that in the above attacks the adversary in fact does not need to create/update fresh values, but simply actively eavesdrop, that is intercept, decompose, compose and copy messages.

Corollary 7. *The existence of a Dolev-Yao attack is undecidable even for bounded memory protocols, $\mathcal{P}_{\mathcal{M}}$, where Alice and Bob are finite automata whom are allowed to update nonces only, all actions by Alice and Bob are balanced, and only terms of height fixed in advance are used by Alice, Bob, and the adversary (even if the actions of the adversary are limited to decompose, compose, and copy).*

Proof. Given a non-recursive recursively enumerable set S, and a sequence of Turing machines $\mathcal{M}_n$ such that $\mathcal{M}_n$ terminates on the empty input iff $n \in S$, it suffices to consider the corresponding bounded memory protocols $\mathcal{P}_{\mathcal{M}_n}$. ■

Thus an upper bound on the memory of the Dolev-Yao adversary is not computable from a bound on the memory used by a protocol. Moreover, based on peculiarities of our encoding described in Section 4.2, we can express such a phenomenon in quantitative terms.

Theorem 8. *Whatever total recursive function h we take, we can construct a recursive sequence of bounded memory protocols $\mathcal{Q}_n$ so that*

(a) *For any n, there is a Dolev-Yao attack on the bounded memory protocol $\mathcal{Q}_n$.*

(b) *However, for any n starting from some n_0, any Dolev-Yao adversary whose memory is bounded by $h(n)$ is not capable of detecting an attack on the bounded memory protocol $\mathcal{Q}_n$.*

Proof Given a total recursive function f, implemented as $\mathcal{Q}_n$ we take the bounded memory protocol $\mathcal{P}_{M_n}$ described in Section 4.2, where M_n is a Turing machine terminating on the empty input with the value $f(n)$.

We assume that h is large enough to be an upper bound for all memories in question.

Given an n, the size of each of the current states of the intruder's memory is supposed to be bounded by $h(n)$, which results in that the number of all different states of his memory times the number of all different states of the memories of the other participants is bounded by $2^{O(h(n))}$. Therefore, interacting with the participants within the protocols $\mathcal{P}_{M_n}$, the bounded memory adversary can only perform at most $2^{O(h(n))}$ steps.

It suffices, therefore, to take a function f such that its time complexity is greater than $2^{O(h(n))}$, for instance, $\Omega(2^{2^{h(n)}})$. Thus, the Dolev-Yao adversary will be able to find an attack, as it can take any number of steps, while the Bounded Memory adversary cannot find an attack. □

The Theorem above implies that the Standard Dolev-Yao adversary cannot be constructively approximated by an infinite sequence of increasing memory Bounded Memory adversaries: for any amount of memory we give to the bounded memory adversary, it is always possible to construct a bounded memory protocol that implements a function that requires more memory.

5. Undecidability Proof Revisited

In this Section we confirm that the secrecy problem is undecidable *even if the memory of the protocol is bounded* in an alternative approach. We revisit the undecidability proof given in [DLMS04] and show that similar encoding can be obtained by using bounded memory protocol theories. This is not obvious since in well-founded protocol theories used in their encoding there is no bound on the memory of the (honest) participants. Namely, an unbounded number of protocol sessions can run concurrently and therefore participants are allowed to remember an unbounded number of facts denoting their participation in protocol sessions and containing the information of the protocol sessions in which they are participating. More formally, while our the rules of our bounded memory protocol theories are all balanced, well-founded protocol theories contain unbalanced rules, which are the source of undecidability of the reachability problem in multiset rewriting systems.

Investigating whether one can adapt the encoding given in [CDL+99, DLMS04] to use bounded memory protocols, instead of well-founded ones was left as future work in our conference paper [KKNS13].

Durgin *et al.* [DLMS04] present an undecidability proof for protocol theories based on the encoding of the existential Horn problem. It is know that the existential Horn problem with no function symbols is undecidable [CLM81, DLMS04]. We now encode the existential Horn problem using bounded memory protocols instead of the well-founded ones used in [DLMS04].

In order to use memory bounded protocols we modify the theories from [DLMS04]. In particular we add role regeneration rules for each of the roles. Protocol theories in [DLMS04] were specified by rules that contained exactly two facts in the pre- and post-condition. Although these rules were balanced, the initialization theory and the role generation theory were unbalanced enabling an unbounded number of protocol sessions. In principle, this allows any number of sessions to run concurrently, which is not possible in bounded memory theories. Nevertheless, there is a run simulating

the derivation in the Horn theory in which only one session runs at any given time. In this run the standard Dolev-Yao adversary with unbounded memory is the initiator of each session. He acts as the network intercepting each message and storing all the intercepted data, while participants only share a secret key and do not need to remember the data from previous sessions.

5.1. Encoding the Existential Horn Problem

An *existential Horn clause* is a formula of the form:

$$\forall x_1 \cdots \forall x_m . \left[\alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \exists y_1 \cdots \exists y_k . \beta_1 \wedge \cdots \wedge \beta_j\right].$$

where $\alpha_1, \cdots \wedge \alpha_n, \beta_1, \cdots \beta_j$ are first-order atomic formulas. The *existential Horn problem* is the problem of determining whether a formula is a consequence of an existential Horn clause theory, *i.e.*, a set $\mathcal{H}$ of existential Horn clauses. Here we are interested in formulas that are conjunctions of atomic first-order logic formulas. For the inference of such a formula ϕ from $\mathcal{H}$, it sufices to check whether ϕ can be obtained by a derivation derivation $\mathcal{H} \vdash \phi$, which is a sequence of formulas $\phi_1, \ldots, \phi_n$ such that $\phi_n = \phi$, and ϕ_i is either

a) $\phi_i \in \mathcal{H}$

b) $\phi_i \equiv \beta_1 \wedge \cdots \wedge \beta_l$ for some $\phi_j \equiv \alpha_1 \wedge \cdots \wedge \alpha_s$, $j < i$ and $\forall \vec{x}[\alpha_1 \wedge \cdots \wedge \alpha_s \Rightarrow \exists \vec{y}.\beta_1 \wedge \cdots \wedge \beta_l] \in \mathcal{H}$

c) $\phi_i \equiv \alpha_k$ for some $\phi_j \equiv \alpha_1 \wedge \cdots \wedge \alpha_k \wedge \cdots \wedge \alpha_s$, $j < i$

d) $\phi_i \equiv \alpha_1 \wedge \cdots \wedge \alpha_l \wedge \cdots \wedge \alpha_m \wedge \cdots \wedge \alpha_s$, for $j_1, \ldots, j_k < i$

$$\begin{gathered} \phi_{j_1} \equiv \alpha_1 \wedge \cdots \wedge \alpha_l \\ \cdots \\ \phi_{j_k} \equiv \alpha_m \wedge \cdots \wedge \alpha_s \\ \text{where } \forall x_1 \cdots \forall x_m . \left[\alpha_1 \wedge \cdots \wedge \alpha_s \Rightarrow \exists \vec{y}.\beta_1 \wedge \cdots \wedge \beta_j\right] \in \mathcal{H} \end{gathered}$$

The above rules correspond to application of a Horn clause from $\mathcal{H}$, conjunction elimination and conjunction introduction, required to transform consequence of one Horn clause into antecedent of another. It follows that, for derivations, the order of atomic formulas on either side of a Horn clause is irrelevant.

Given a set of existential Horn clauses $\mathcal{H}$, we construct a bounded memory protocol theory and define representations of formulas as encodings of formulas given by (4) below. For such protocol theory, when combined with the standard Dolev-Yao adversary theory given in Figure 1a, the adversary may learn the representation of a formula if and only if that formula is a consequence of the given Horn theory $\mathcal{H}$.

To each Horn clause X from the given theory $\mathcal{H}$ we associate a number of protocol roles, $\mathcal{P}(\mathcal{H})$. Namely, to the clause

$$X \equiv \forall \vec{x}[\alpha_1 \wedge \cdots \wedge \alpha_m \Rightarrow \exists \vec{y}.\beta_1 \wedge \cdots \wedge \beta_i \wedge \cdots \wedge \beta_l]$$

correspond the role theories $\mathcal{R}(X)$, $\mathcal{D}^i(X)$ for $i = 1, \ldots, l$, $\mathcal{C}(X)$ and the role regeneration theory $\mathcal{G}(X)$, defined below. Role theory $\mathcal{R}(X)$ corresponds to the application of Horn clause X. Each role theory $\mathcal{D}^i(X)$ corresponds to conjunction elimination rule, needed to extract atomic formulas from the conjunctions. Role theory $\mathcal{C}(X)$ corresponds to conjunction introduction, allowing the set of atomic formulas to be combined to produce the conjunction needed to apply another Horn clause. Formally, we define these role theories for X as follows:

$$\mathcal{R}(X) : \quad A_0^X(k)\ N_R(\lceil \alpha_1 \wedge \cdots \wedge \alpha_m \rceil) \rightarrow \exists \vec{y}. A_1^X(k)\ N_S(\lceil \beta_1 \wedge \cdots \wedge \beta_l \rceil)$$

$$\mathcal{D}^i(X) : \quad B_0^{X,i}(k)\ N_R(\lceil \beta_1 \wedge \cdots \wedge \beta_i \wedge \cdots \wedge \beta_l \rceil) \rightarrow B_1^{X,i}(k)\ N_S(\lceil \beta_i \rceil)\ , \quad i = 1, \ldots, l$$

$$\begin{aligned}
\mathcal{C}(X) : \quad & C_0^X(k)\ N_R(\lceil \alpha_1 \rceil) \rightarrow C_1^X(k, \lceil \alpha_1 \rceil)\ N_S(\top) \\
& C_1^X(k, \lceil \alpha_1 \rceil)\ N_R(\lceil \alpha_2 \rceil) \rightarrow C_2^X(k, \lceil \alpha_1 \wedge \alpha_2 \rceil)\ N_S(\top) \\
& \quad \cdots \\
& C_{i-1}^X(k, \lceil \alpha_1 \wedge \cdots \wedge \alpha_{i-1} \rceil)\ N_R(\lceil \alpha_i \rceil) \rightarrow C_2^X(k, \lceil \alpha_1 \wedge \cdots \wedge \alpha_i \rceil)\ N_S(\top) \\
& \quad \cdots \\
& C_{m-1}^X(k, \lceil \alpha_1 \wedge \cdots \wedge \alpha_{m-1} \rceil)\ N_R(\lceil \alpha_k \rceil) \rightarrow C_m^X(k)\ N_S(\lceil \alpha_1 \wedge \cdots \wedge \alpha_m \rceil)
\end{aligned}$$

$$\begin{aligned}
\mathcal{G}(X) : \quad & Guy(g,k)\ P(*) \rightarrow \ Guy(g,k)\ A_0^X(k) \\
& A_1^X(k) \rightarrow \ P(*) \\
\\
& Guy(k)\ P(*) \rightarrow \ Guy(k)\ B_0^{X,i}(k) \\
& B_1^{X,i}(k) \rightarrow \ P(*)\ , \qquad i = 1, \ldots, l \\
\\
& Guy(g,k)\ P(*) \rightarrow \ Guy(g,k)\ C_0^X(k) \\
& C_m^X(k) \rightarrow \ P(*)
\end{aligned}$$

where the term $\top$ that represents "true", *i.e.* the empty conjunction, and $\lceil \phi \rceil$ denotes the encoding of a formula ϕ of the form

$$P_1(t_{1,1}, \ldots, t_{1,i_1}) \wedge \cdots \wedge P_j(t_{j,1}, \ldots, t_{j,i_j})$$

into a term of type message as specified below:

$$\lceil \phi \rceil = \ E_k(\langle P_1.P_2.\ \ldots\ .P_j, t_{1,1}, \ldots, t_{1,i_1}, \ldots, t_{j,1}, \ldots, t_{j,i_j} \rangle)\ . \tag{4}$$

Here we assume that for each sequence of predicates $P_1, \ldots, P_j$ that occurs on either side of the given Horn clauses in $\mathcal{H}$ there is a constant symbol $P_1.P_2.\ \ldots\ .P_j$ in the signature. In the above representation, encryption under the secret key k not known to the adversary is needed for a faithful encoding of the theory. Otherwise the adversary would be able to interfere with the atomic formulas.

However, intruder needs to extract atomic formulas from conjunctions. That is, consequents of Horn clauses must be decomposed into their constituent atomic formulas, so that they can then be combined into antecedents of the next Horn clause in the derivation. We therefore add to $\mathcal{P}(\mathcal{H})$ role theory $\mathcal{R}(X_m^i)$, and the role regeneration theory $\mathcal{G}(X_m^i)$ for each of the following clauses:

$$X_m^i \equiv \forall \vec{x}[\alpha_1 \wedge \cdots \wedge \alpha_i \wedge \cdots \wedge \alpha_m \Rightarrow \alpha_i], \ \ i = 1, \ldots, m$$

where $m = 2, \ldots, b$ and b is the maximal number of atomic formulas in a consequent of a clause in $\mathcal{H}$. This way we obtain the following theories:

$$\mathcal{R}(X^i_m) \ : A_0^{X^i_m}(k)\ N_R(\lceil \alpha_1 \wedge \cdots \wedge \alpha_m \rceil) \rightarrow A_1^{X^i_m}(k)\ N_S(\lceil \alpha_i \rceil)$$

and

$$\begin{array}{rrl} \mathcal{G}(X^i_m) \ : & Guy(g,k)\ P(*) \rightarrow & Guy(g,k)\ A_0^{X^i_m}(k) \\ & A_1^{X^i_m}(k) \rightarrow & P(*) \end{array}$$

We will now show that there is a connection between intruder knowledge of representations of formulas and derivations of these formulas in the Horn theory. As usual, terms in M-facts represent the knowledge of the adversary. In particular, fact $M(\lceil \phi \rceil)$ will denote that adversary has learnt the representation of formula ϕ, that is, $\lceil \phi \rceil$ is stored in his memory.

Let the initial configuration contain the following facts:

$$Guy(A,K),\ Guy(B,K), M(\top),\ P(*),\ P(*)$$

Facts $Guy(A,K),\ Guy(B,K)$ denote that Alice and Bob share the uncompromised key K. Fact $M(\top)$ is used by the adversary to send empty conjunctions when needed. One empty fact $P(*)$ serves for the network, *i.e.* for running the protocol. The second empty fact $P(*)$ will be used by the role regeneration theory for the role states of the single running session. It will be replaced by an empty fact when the session terminates. Notice that all the role theories are responder roles and that the session runs between an honest participant and the adversary who is the initiator of all sessions.

Theorem 9. *Let $\mathcal{P}(\mathcal{H})$ be the encoding of the Horn Theory $\mathcal{H}$ into the bounded memory protocols as described above. The standard Dolev-Yao adversary can learn the representation of a formula ϕ from a run of protocols $\mathcal{P}(\mathcal{H})$ with the initial configuration $Guy(A,K),\ Guy(B,K),\ M(\top),\ P(*),\ P(*)$ if and only if ϕ is derivable from $\mathcal{H}$.*

Proof We modify the proof from [DLMS04] to accommodate bounded memory protocols.

We first show that if $\mathcal{H} \vdash \phi$ then $M(\lceil \phi \rceil)$ appears in adversary's memory. The proof is by induction on the length of derivations. For the base case we consider a formula $\phi \in \mathcal{H}$, that is the clause $X \equiv true \Rightarrow \phi$ from $\mathcal{H}$, the corresponding protocol role

$$\mathcal{R}(X) :\ A_0^X(k)\ N_R(\top) \rightarrow A_1^X(k)\ N_S(\lceil \phi \rceil)$$

and the role regeneration theory $\mathcal{G}(X)$. The adversary can obtain the fact $M(\lceil \phi \rceil)$ from the following run with either Alice or Bob:

$$\begin{array}{rcl} Guy(g,k)\ P(*) & \rightarrow & Guy(g,k)\ A_0^X(k) \\ M(\top) & \rightarrow & N_R(\top) \\ A_0^X(k)\ N_R(\top) & \rightarrow & A_1^X(k)\ N_S(\lceil \phi \rceil) \\ N_S(\lceil \phi \rceil) & \rightarrow & M(\lceil \phi \rceil) \end{array} \tag{5}$$

For the induction step, we look at the derivation $\mathcal{H} \vdash \phi$ of length n where the last formula $\phi_n \equiv \phi$. We consider the possible cases, as per derivation definition. If $\phi \in \mathcal{H}$, adversary can obtain the fact $M(\lceil\phi\rceil)$ as in the base case above.

Let ϕ be the result of applying the Horn clause $X \equiv \alpha_1 \wedge \cdots \wedge \alpha_s \Rightarrow \exists \vec{z}.\phi(\vec{z})$ to a formula $\phi_j \equiv \alpha_1 \wedge \cdots \wedge \alpha_s$, where $j < i$. From the inductive hypothesis applied to the derivation $\mathcal{H} \vdash \phi_j$ of lower length j, the adversary knows the representation of ϕ_j, *i.e.* $M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)$ appears in adversary's memory. From the following consecutive run of roles $\mathcal{R}(X_s^1)$, ..., $\mathcal{R}(X_s^s)$, adversary learns representations of atomic formulas of ϕ_j.

$$
\begin{aligned}
Guy(g,k)\ P(*) &\rightarrow Guy(g,k)\ A_0^{X_s^1}(k)\\
M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
A_0^{X_s^1}(k)\ N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow A_1^{X_s^1}(k)\ N_S(\lceil\alpha_1\rceil)\\
N_S(\lceil\alpha_1\rceil) &\rightarrow M(\lceil\alpha_1\rceil)\\
A_1^{X_s^1}(k) &\rightarrow P(*)\\
Guy(g,k)\ P(*) &\rightarrow Guy(g,k)\ A_0^{X_s^2}(k)\\
M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
A_0^{X_s^2}(k)\ N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow A_1^{X_s^2}(k)\ N_S(\lceil\alpha_2\rceil)\\
N_S(\lceil\alpha_2\rceil) &\rightarrow M(\lceil\alpha_2\rceil)\\
A_1^{X_s^2}(k) &\rightarrow P(*)\\
&\cdots\\
Guy(g,k)\ P(*) &\rightarrow Guy(g,k)\ A_0^{X_s^s}(k)\\
M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
A_0^{X_s^s}(k)\ N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow A_1^{X_s^s}(k)\ N_S(\lceil\alpha_s\rceil)\\
N_S(\lceil\alpha_s\rceil) &\rightarrow M(\lceil\alpha_s\rceil)
\end{aligned}
$$

Hence, facts $M(\lceil\alpha_1\rceil), \ldots, M(\lceil\alpha_s\rceil)$ appear in adversary's memory.

Now, from the below run, the adversary is able to learn $M(\lceil\phi\rceil)$:

$$
\begin{aligned}
Guy(g,k)\ P(*) &\rightarrow Guy(g,k)\ C_0^X(k)\\
M(\lceil\alpha_1\rceil) &\rightarrow N_R(\lceil\alpha_1\rceil)\\
C_0^X(k)\ N_R(\lceil\alpha_1\rceil) &\rightarrow C_1^X(k, \lceil\alpha_1\rceil)\ N_S(\top)\\
N_S(\top) &\rightarrow M(\top)\\
M(\lceil\alpha_2\rceil) &\rightarrow N_R(\lceil\alpha_2\rceil)\\
C_1^F(k, \lceil\alpha_1\rceil)\ N_R(\lceil\alpha_2\rceil) &\rightarrow C_2^F(k, \lceil\alpha_1 \wedge \alpha_2\rceil)\ N_S(\top)\\
N_S(\top) &\rightarrow M(\top)\\
M(\lceil\alpha_3\rceil) &\rightarrow N_R(\lceil\alpha_3\rceil)\\
&\cdots\\
M(\lceil\alpha_s\rceil) &\rightarrow N_R(\lceil\alpha_s\rceil)\\
C_{s-1}^X(k, \lceil\alpha_1 \wedge \cdots \wedge \alpha_{s-1}\rceil)\ N_R(\lceil\alpha_s\rceil) &\rightarrow C_s^X(k)\ N_S(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
C_s^X(k) &\rightarrow P(*)\\
Guy(g,k)\ P(*) &\rightarrow Guy(g,k)\ A_0^X(k)\\
N_S(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
M(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil)\\
A_0^X(k)\ N_R(\lceil\alpha_1 \wedge \cdots \wedge \alpha_s\rceil) &\rightarrow A_1^X(k)\ N_S(\lceil\phi\rceil)\\
N_S(\lceil\phi\rceil) &\rightarrow M(\lceil\phi\rceil)
\end{aligned}
$$

The above run contains the roles $\mathcal{R}(X)$, $\mathcal{C}(X)$ and the role regeneration theory $\mathcal{G}(X)$ corresponding to clause X.

Notice that there is enough memory in the system so that all the above rules are applicable. More precisely, one empty fact is used for the network and the other for role state predicates. Since only one role is active at a time that is enough memory for the whole run.

The case when ϕ is obtained by the conjunction elimination rule, $\phi = \alpha_i$, reduces to the above case for the clause X^i_m.

For the remaining case of the application of conjunction introduction rule to obtain ϕ, it follows that ϕ is the antecedent of some clause $X \in \mathcal{H}$. Adversary can therefore learn the representation of ϕ by initiating a role $\mathcal{C}(X)$ corresponding to that clause.

Next we show that if from a run of protocol theories $\mathcal{P}(\mathcal{H})$ the adversary learns $M(\lceil\phi\rceil)$ then $\mathcal{H} \vdash \phi$. The proof is by induction on the length of the run, *i.e.*, the number of roles. Since we assume that the key K is uncompromised, initially no facts $M(\lceil\phi\rceil)$ appear in adversary's memory. The adversary can only learn $M(\lceil\phi\rceil)$ from some message sent by the rules in $\mathcal{P}(\mathcal{H})$. The shortest such run is similar to the sequence of rules (5) and corresponds to the clause $true \Rightarrow \phi$ from $\mathcal{H}$. Trivially it is the case that $\mathcal{H} \vdash \phi$.

We then consider the run of n protocol roles from $\mathcal{P}(\mathcal{H})$ in which the adversary learns $M(\lceil\phi\rceil)$. Since the system's memory is bounded, only one role can run at a time. Role state predicates of previous session have been forgotten by the role regeneration rules and the last role in the run, $\mathcal{R}_n$, has started executing. We assume that after the first $n-1$ roles the adversary knows $\{M(\lceil\phi_1\rceil), \ldots, M(\lceil\phi_k\rceil)\}$. By the induction hypothesis we have $\mathcal{H} \vdash \{\phi_1, \ldots, \phi_k\}$. As per the construction of protocol theories $\mathcal{P}(\mathcal{H})$ from the Horn theory $\mathcal{H}$, role $\mathcal{R}_n$ is either $\mathcal{R}(X)$, $\mathcal{D}^i(X)$, $\mathcal{C}(X)$ or $\mathcal{G}(X)$ for some clause $X \in \mathcal{H}$, or $X \equiv X^i_m$, for $X^i_m \equiv \forall\vec{x}[\alpha_1 \wedge \cdots \wedge \alpha_i \wedge \cdots \wedge \alpha_m \Rightarrow \alpha_i]$. If the role $\mathcal{R}_n$ is a $\mathcal{R}(X)$ role, then X is the clause $\phi_i \Rightarrow \exists\vec{y}\phi$ for some $i \in \{1, \ldots, k\}$ and because of $\mathcal{H} \vdash \phi_i$ we have $\mathcal{H} \vdash \phi$.

Roles $\mathcal{D}^i(X)$ and $\mathcal{R}(X^i_m)$ correspond to the logical axiom

$$A_1 \wedge \cdots \wedge A_i \wedge \cdots \wedge A_j \Rightarrow A_i\,.$$

This role represents the application of above axiom to a formula from $\{\phi_1, \ldots, \phi_k\}$ which results in ϕ. From $\mathcal{H} \vdash \{\phi_1, \ldots, \phi_k\}$ and the above axiom it follows that $\mathcal{H} \vdash \phi$.

If the role $\mathcal{R}_n$ is a $\mathcal{C}(X)$ role and all its rules have been used in the run, then F is the logical axiom

$$A_1 \wedge \cdots \wedge A_i \Rightarrow (A_1 \wedge \cdots \wedge A_i)$$

for some $i \in \{1, \ldots, k\}$, where a number of formulas implies their conjunction. In case the role $\mathcal{C}(X)$ has not finished, *i.e.* the last rule of the role has not been used in the run, and similarly in the case of role $\mathcal{R}_n$ being $\mathcal{G}(X)$ for some X, no new representations of formulas are exchanged. Consequently $M(\lceil\phi\rceil) \in \{M(\lceil\phi_1\rceil), \ldots, M(\lceil\phi_k\rceil)\}$ and hence $\mathcal{H} \vdash \{\phi\}$. $\square$

Notice that in the above interaction with the protocol run the adversary does not need to create fresh values. He only needs to intercept and send messages.

The next Lemma shows that the encoding of the given Horn theory into protocol theories is polynomial, and that the size of the facts in the obtained protocol theories is linearly bounded.

Lemma 10. *The construction of protocol theories $\mathcal{P}(\mathcal{H})$ from the Horn theory $\mathcal{H}$ is computable in polynomial time. Furthermore, if the formulas in $\mathcal{H}$ have the maximum term size bounded by s, then the size of facts appearing in the run of $\mathcal{P}(\mathcal{H})$ from which the adversary can learn the representation of a formula $\phi \in \mathcal{H}$ is bounded by $f(s)$ where f is a linear function of s.*

Proof Let $\mathcal{H} = \{F_1, \ldots, F_n\}$ be a Horn theory and let m be the bound on the number of atomic formulas in a conjunction in any of the clauses from $\mathcal{H}$. To each clause in $\mathcal{H}$ correspond one $\mathcal{R}(F)$, one $\mathcal{C}(F)$, one $\mathcal{G}(F)$ theories, and up to m $\mathcal{D}^i(F)$ theories. Each theory $\mathcal{R}(F)$ and $\mathcal{D}^i(F)$ has a single rule, each theory $\mathcal{C}(F)$ has up to m rules while each theory $\mathcal{G}(F)$ has a maximum of $2m + 4$ rules. Therefore the construction of protocol theories $\mathcal{P}(\mathcal{H})$ is polynomial in n and m.

Consider a run of $\mathcal{P}(\mathcal{H})$ from which the adversary learns the representation of a formula $\phi \in \mathcal{H}$. Let s be the maximum term size appearing in the Horn theory $\mathcal{H}$, where we count one for each predicate symbol, each term symbol and each conjunction. Recall that the size of a fact is the total number of term and predicate symbols it contains. As per the construction of $\mathcal{P}(\mathcal{H})$ and the run in which the adversary learns $\lceil\phi\rceil$ (see proof of Theorem 9), facts of the largest size have the form of a predicate over the representation of a formula from $\mathcal{H}$, *e.g.*, $C^F_{k-1}(\lceil\phi_1 \wedge \cdots \wedge \phi_k\rceil)$, $N_S(\lceil\phi_1 \wedge \cdots \wedge \phi_k\rceil)$ or $M(\lceil\phi_1 \wedge \cdots \wedge \phi_k\rceil)$. By Definition 4 the number of predicate and term symbols in a representation of a formula from $\mathcal{H}$ is at most $s + 2$, when counting the key and the encryption. Therefore the size of facts appearing in the run is bounded by $s + 3$, *i.e.* it is linearly bounded with respect to s. □

Durgin *et al.* [DLMS04] show that the existential Horn problem is undecidable even when no function symbols are allowed. The following result is a direct consequence of this fact and of Theorem 9.

Corollary 11. *The existence of a Dolev-Yao attack is undecidable even for bounded memory protocols, where only terms of height fixed in advance are used by participants and the adversary (even if the actions of the adversary are limited to intercept messages, copy and send).*

6. Related Work

This paper strengthens the undecidability proof given in [CDL+99, DLMS04]. In particular, the proof in [CDL+99, DLMS04] uses an encoding with well-founded protocol theories, whereas our proof uses an encoding with bounded memory protocols. While in bounded memory protocols the memory of the honest participants is bounded, in well-founded protocols it is possible for the honest participants to have an unbounded memory. This is in fact the case in the undecidability proof given in [CDL+99, DLMS04]. The proof relies on an unbounded number of protocol sessions. Moreover, all these protocols sessions are created before any sessions starts executing.

Hence participants require an unbounded memory to remember in which protocol sessions they are participating. On the other hand, in our proof, Alice and Bob participate in one protocol session at a time. Whenever one is finished, they re-use their memory to participate in the subsequent protocol session. This difference is crucial, as with our proof, we can infer that there is no way to compute an upper bound on the memory of the adversary from the memory bounds of the participants, demonstrating further the hardness of the secrecy problem.

Our paper is closely related to frameworks based on multiset rewriting systems used to specify and verify security properties of protocols [AL00, ALV03, CKRT05, CLS03, DLMS04, RT03]. Assumptions used by these frameworks reflect open systems where an intruder tries to attack the participants of the system by manipulating the transmitted messages. The related security problems consider a powerful intruder that has an unbounded memory and that can, for example, accumulate messages at will. As well as considering such an intruder, in addition we study the intruder with bounded memory. With our bounded memory intruder, we can imagine having a system where (honest and dishonest) agents are in a *closed room* and collaborate. We assume that all agents, including inside adversaries, have bounded memory, technically imposed by the use of balanced actions and facts of bounded size. Moreover, we compare the strengths of these intruder models in relation to the secrecy problem with bounded memory protocols.

The complexity results for protocol insecurity obtained in [ALV03, RT03] assume no bound on the size of adversary messages. The proofs rely exactly on the nature of adversary rules, namely composition and decomposition rules, and do not apply to general rewriting systems. In our bounded memory systems and bounded memory adversaries we do assume a bound on the size of facts. This condition normally appears for example, in the specification of administrative processes, where only tokens are used and no function symbols. [KKN+12]. Although lifting this bound could make sense if no function symbols were allowed, the bound on the depth of terms is deeply embedded in the semantics of our balanced systems. This bound is needed for our [KKNS] PSPACE-complete complexity result for the reachability problem for balanced systems. Moreover, it was shown in [DLMS04] that, with no bound on the size of facts, the reachability problem is undecidable even if the system is balanced.

In this paper, we do not make any assumptions on protocols nor on the format of the exchanged messages. We only bound the size of messages exchanged, number of concurrent protocol sessions and the amount of memory of the bounded memory intruder. Even restricting to balanced protocol theories is not an assumption on protocols, as by using empty facts, it is possible to transform a unbalanced protocol theory into a balanced one. It is possible, however, to recover the decidability of the secrecy problem if further assumptions on the protocol are made even with an unbounded memory intruder and with unbounded number of parallel protocol sessions. For instance, [RS03] shows that for protocols tagging mechanisms, the secrecy problem is decidable. [CWZ07] proposes a general technique to construct safe protocols by using digital signatures linked to protocol sessions. Also [ADK08] proposes a general technique to construct a secure protocol by using both digital signature and dynamic tagging mechanisms.

Harrison *et al.* present a formal approach to access control [HRU75] and faithfully encode a Turing machine in their system. However, in contrast to our encoding, they

use a non-commutative matrix to encode the sequential, non-commutative tape of a Turing machine. In their proofs, the non-commutative nature of the encoding plays an important role. We, on the other hand, encode Turing machine tapes by using commutative multisets. Specifically, they show that if no restrictions are imposed to the systems, the reachability problem is undecidable.

Much work on reachability related problems has been done within the Petri nets community, see *e.g.*, [EN94]. Specifically, we are interested in the *coverability problem* which is closely related to the reachability problem in multiset rewrite systems. To the best of our knowledge, no work that captures exactly the balanced condition has yet been proposed. It does not seem possible to provide direct, *faithful* reductions between our balanced systems and Petri nets.

7. Conclusions

This paper shows that the memory of the adversary cannot be inferred from the memory bounds of the participants (Theorem 8). This is accomplished by proposing a novel undecidability proof by encoding Turing machines by means of bounded memory protocols. This result confirms the hardness of protocol security. It answers negatively an open problem left in [KKNS]. We further confirm the undecidability of the secrecy problem for bounded memory protocols and the standard Dolev-Yao adversary by revisiting the encoding the existential Horn implication problem shown in [CDL+99, DLMS04] and demonstrating that this problem can also be encoded by means of Bounded Memory Protocols.

Together with Carolyn Talcott, we are investigating the use of the computational tool Maude [CDE+07] for the specification and model-checking of regulated processes, such as administrative processes [KKN+12].

Another direction that we are currently investigating is the extension of our model with continuous time. In particular, systems that can create fresh values and mention continuous time are of great interest to protocol security. For instance, many distance authentication protocols [MPP+07, BC94] rely on timing measures. Thus extending our model with continuous time and determining decidable fragments, *e.g.*, balanced systems, is of great interest for the verification of such protocols.

Acknowledgments: We thank Elie Bursztein, Iliano Cervesato, Patrick Lincoln, Joshua Guttman, Catherine Meadows, Dale Miller, John Mitchell, Paul Rowe, and Carolyn Talcott for helpful discussions. This material is based upon work supported by the MURI program under AFOSR Grant No: FA9550-08-1-0352 and upon work supported by the AFOSR MURI "Science of Cyber Security: Modeling, Composition, and Measurement". Additional support for Scedrov from NSF Grant CNS-0830949 and from ONR grant N00014-11-1-0555. Nigam was partially supported by the Alexander von Humboldt Foundation and CNPq. Kanovich was partially supported by the EPSRC.

[ADK08] Myrto Arapinis, Stéphanie Delaune, and Steve Kremer. From one session to many: Dynamic tags for security protocols. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, LPAR '08, pages 128–142, Berlin, Heidelberg, 2008. Springer-Verlag.

[AL00] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *CONCUR '00: Proceedings of the 11th International Conference on Concurrency Theory*, pages 380–394, London, UK, 2000. Springer-Verlag.

[ALV03] Roberto M. Amadio, Denis Lugiez, and Vincent Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theor. Comput. Sci.*, 290(1):695–740, 2003.

[BC94] Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 344–359, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[CDE+07] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude: A High-Performance Logical Framework*. Springer, 2007.

[CDL+99] Iliano Cervesato, Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A meta-notation for protocol analysis. In *CSFW*, pages 55–69, 1999.

[CKRT05] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP decision procedure for protocol insecurity with XOR. *Theor. Comput. Sci.*, 338(1-3):247–274, 2005.

[CLM81] Ashok K. Chandra, Harry R. Lewis, and Johann A. Makowsky. Embedded implicational dependencies and their inference problem. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, STOC '81, pages 342–354, New York, NY, USA, 1981. ACM.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS '03: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, page 271, Washington, DC, USA, 2003. IEEE Computer Society.

[CWZ07] Véronique Cortier, Bogdan Warinschi, and Eugen Zalinescu. Synthetizing secure protocols. In Joachim Biskup and Javier Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 2007.

[DLMS04] Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[EN94] Javier Esparza and Mogens Nielsen. Decidability issues for Petri nets - a survey. *Bulletin of the EATCS*, 52:244–262, 1994.

[HRU75] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. On protection in operating systems. In *SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles*, pages 14–24, New York, NY, USA, 1975. ACM.

[KKN+12] Max I. Kanovich, Tajana Ban Kirigin, Vivek Nigam, Andre Scedrov, Carolyn L. Talcott, and Ranko Perovic. A rewriting framework for activities subject to regulations. In Ashish Tiwari, editor, *RTA*, volume 15 of *LIPIcs*, pages 305–322. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[KKNS] Max Kanovich, Tajana Ban Kirigin, Vivek Nigam, and Andre Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. *Inf. Comput.* Accepted for Publication.

[KKNS13] Max I. Kanovich, Tajana Ban Kirigin, Vivek Nigam, and Andre Scedrov. Bounded memory protocols and progressing collaborative systems. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2013.

[KRS09] Max Kanovich, Paul Rowe, and Andre Scedrov. Policy compliance in collaborative systems. In *CSF '09: Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, pages 218–233, Washington, DC, USA, 2009. IEEE Computer Society.

[KRS11] Max I. Kanovich, Paul Rowe, and Andre Scedrov. Collaborative planning with confidentiality. *J. Autom. Reasoning*, 46(3-4):389–421, 2011.

[MPP+07] Catherine Meadows, Radha Poovendran, Dusko Pavlovic, LiWu Chang, and Paul F. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In Radha Poovendran, Sumit Roy, and Cliff Wang, editors, *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30 of *Advances in Information Security*, pages 279–298. Springer, 2007.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[RS03] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science (FST TCS 2003), volume 2914 of Lecture Notes in Computer Science*, pages 363–374. Springer, 2003.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete. *Theor. Comput. Sci.*, 299(1-3):451–475, 2003.