Journal of Computer Security vol 16, no 6, 689-712, 2008.

Secrecy for Bounded Security Protocols With Freshness Check is NEXPTIME-complete *

Ferucio L. Ţiplea [a),b)], Cătălin V. Bîrjoveanu [a)],
Constantin Enea [a)], and Ioana Boureanu [a)]

[a)]Department of Computer Science, "Al.I.Cuza" University of Iaşi
Iaşi, Romania, E-mail: {cbirjoveanu,cenea,iboureanu}@infoiasi.ro

[b)] School of Computer Science, University of Central Florida
Orlando, FL 32816, USA, E-mail: fltiplea@mail.dntis.ro

Abstract

The *secrecy problem* for security protocols is the problem to decide whether or not a given security protocol has leaky runs. In this paper, the (initial) secrecy problem for bounded protocols with freshness check is shown to be NEXPTIME-complete. Relating the formalism in this paper to the multi-set rewriting (MSR) formalism we obtain that the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, with or without disequality tests, and an intruder with no existentials, is NEXPTIME-complete. If existentials for the intruder are allowed but disequality tests are not allowed, the initial secrecy problem still is NEXPTIME-complete. However, if both existentials for the intruder and disequality tests are allowed and the protocols are not well-founded (and, therefore, not in restricted form), then the problem is undecidable. These results also correct some wrong statements in [Durgin et al., JCS 2004].

Key words: security protocol, secrecy problem, complexity

*The research reported in this paper was partially supported by the National University Research Council of Romania under the grant CNCSIS 632/2004-2005.

1 Introduction and Preliminaries

A security protocol specifies a set of rules under which a sequence of messages is exchanged between two or more parties in order to achieve security goals such as authentication or establishing new shared secrets. Examples of security protocols include the Kerberos authentication scheme used to manage encrypted passwords on clusters of interconnected computers, Secure Sockets Layer used by Internet browsers and servers to carry out secure Internet transactions etc.

One of the main questions one may ask when dealing with security protocols is the following: How difficult to prove is that a security protocol assures secrecy? Secrecy is undecidable in general but, if security protocols are restricted by bounding the number of nonces, the message lengths, the number of sessions etc., decidability of secrecy can be gained (see [4] for more details).

Secrets in a security protocol can be provided from the beginning or in the course of runs. When they are provided from the beginning, the secrecy problem is called the *initial secrecy problem*. Using a multiset rewriting (MSR) formalism, it has been claimed in [4] (Table 9, page 282) that the following problems are DEXPTIME-complete:

- the initial secrecy problem for security protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials;
- the initial secrecy problem for security protocols in restricted form, with bounded length messages, bounded existentials, disequality tests, and an intruder with no existentials;
- the initial secrecy problem for security protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials.

Unfortunately, all these statements are wrong. The DEXPTIME-completeness result holds true only for the case with "no existentials" and "an intruder with no existentials" as it was correctly stated in [3] (Section 2.1 provides full details about this case).

Using a formalism and a terminology slightly different than the one in [4], we show that the (initial) secrecy problem for bounded security protocols with freshness check is NEXPTIME-complete (Section 2.2). Relating the formalism in this paper to the MSR formalism we obtain that the three problems mentioned above are NEXPTIME-complete (Section 2.3). Moreover, the initial secrecy problem for security protocols in restricted form, with bounded length messages, bounded existentials, disequality

tests, and an intruder with bounded existentials, is NEXPTIME-complete too. If "an intruder with bounded existentials" is replaced by "an intruder with existentials" and security protocols are not well-founded (and, therefore, not in restricted form), this last problem becomes undecidable. Therefore, the problem marked "???" in Table 9 of [4] still remains open.

In what follows we will adopt the formalism proposed in [5] with slight modifications, and we will use it in order to develop the main result of the paper.

Protocol signatures and terms. A *security protocol signature* is a 3-tuple $\mathcal{S} = (\mathcal{A}, \mathcal{K}, \mathcal{N})$ consisting of a finite set $\mathcal{A}$ of *agent names* (or shortly, *agents*) and two at most countable sets $\mathcal{K}$ and $\mathcal{N}$ of *keys* and *nonces*, respectively. It is assumed that:

- $\mathcal{A}$ contains a special element denoted by I and called the *intruder*. All the other elements are called *honest agents* and Ho denotes their set;
- $\mathcal{K} = \mathcal{K}_0 \cup \mathcal{K}_1$, where $\mathcal{K}_0$ is the set of *short-term keys* and $\mathcal{K}_1$ is a finite set of *long-term keys*. The elements of $\mathcal{K}_1$ are of the form K_A^e (A's public key), or K_A^d (A's private key), or K_{AB} (shared key by A and B), where A and B are distinct agents. $\mathcal{K}_A$ denotes the set of long-term keys known by $A \in \mathcal{A}$;
- some honest agents are provided from the beginning with some *secret information*, not known to the intruder. Denote by $Secret_A \subseteq \mathcal{K}_0 \cup \mathcal{N}$ the set of secret information the honest agent A is provided from the beginning. $Secret_A$ does not contain long-term keys because they will never be communicated by agents during the runs;
- the intruder is provided from the beginning with a set of nonces $\mathcal{N}_I \subseteq \mathcal{N}$ and a set of short-term keys $\mathcal{K}_{0,I} \subseteq \mathcal{K}_0$. It is assumed that no elements in $\mathcal{N}_I \cup \mathcal{K}_{0,I}$ can be generated by honest agents.

The set of *basic terms* is $\mathcal{T}_0 = \mathcal{A} \cup \mathcal{K} \cup \mathcal{N}$. The set $\mathcal{T}$ of *terms* is defined inductively by: every basic term is a term; if t_1 and t_2 are terms, then (t_1, t_2) is a term; if t is a term and K is a key, then $\{t\}_K$ is a term. We extend the construct (t_1, t_2) to $(t_1, \ldots, t_n)$ as usual by letting $(t_1, \ldots, t_n) = ((t_1, \ldots, t_{n-1}), t_n)$, for all $n \geq 3$. Sometimes, parenthesis will be omitted. Given a term t, $Sub(t)$ is the set of all *subterms* of t (defined as usual). This notation is extended to sets of terms by union.

The length of a term is defined as usual, by taking into consideration that pairing and encryption are operations. Thus, $|t| = 1$ for any $t \in \mathcal{T}_0$, $|(t_1, t_2)| = |t_1| + |t_2| + 1$, for any terms t_1 and t_2, and $|\{t\}_K| = |t| + 2$, for any term t and key K.

The *perfect encryption assumption* we adopt [2] states that a message encrypted with a key K can be decrypted only by an agent who knows the corresponding inverse of K (denoted K^{-1}), and the only way to compute $\{t\}_K$ is by encrypting t with K.

Actions. There are two types of actions, send and receive. A *send action* is of the form $A!B : (M)t$, while a *receive action* is of the form $A?B : t$. In both cases, A is assumed an honest agent, $A \neq B$, $t \in \mathcal{T}$ is the *term of the action*, and $M \subseteq Sub(t) \cap (\mathcal{N} \cup \mathcal{K}_0)$ is the *set of new terms of the action*.

$M(a)$ denotes the set M, if $a = A!B : (M)t$, and the empty set, if $a = A?B : t$; $t(a)$ stands for the term of a. When $M = \emptyset$ we will simply write $A!B : t$. The set of all actions performed by A is $Act(A) = \{A!B : (M)t | B \in \mathcal{A}\} \cup \{A?B : t | B \in \mathcal{A}\}$. For a sequence of actions $w = a_1 \cdots a_l$ and an agent A, we define the *restriction of w to A* as being the sequence obtained from w by removing all actions not in $Act(A)$. Denote this sequence by $w|_A$. The notations $M(a)$ and $t(a)$ are extended to sequences of actions by union.

Protocols. A *security protocol* (or simply, *protocol*) is a triple $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$, where $\mathcal{S}$ is a security protocol signature, $\mathcal{C}$ is a subset of $\mathcal{T}_0$, called the set of *constants* of $\mathcal{P}$, and w is a non-empty sequence of actions, called the *body* of the protocol, such that no action in w contains the intruder. Constants are publicly known elements in the protocol that cannot be re-instantiated (as it will be explained below). As usual, $\mathcal{C}$ does not include private keys, elements in $Secret_A$ for any honest agent A, or elements in $\mathcal{N}_I$, $\mathcal{K}_{0,I}$ and $M(w)$.

Any non-empty sequence $w|_A$, where A is an agent, is called a *role* of the protocol. A role specifies the actions a participant should perform in a protocol, and the order of these actions.

Substitutions and events. Instantiations of a protocol are given by *substitutions*, which are functions σ that map agents to agents, nonces to arbitrary terms, short-term keys to short-term keys, and long-term keys to long-term keys. Moreover, for long-term keys, σ should satisfy $\sigma(K_A^e) = K_{\sigma(A)}^e$, $\sigma(K_A^d) = K_{\sigma(A)}^d$, and $\sigma(K_{AB}) = K_{\sigma(A)\sigma(B)}$, for any distinct agents A and B.

Substitutions are homomorphically extended to terms, actions, and sequences of actions. A substitution σ is called *suitable for an action $a = AxB : y$* if $\sigma(A)$ is an honest agent, $\sigma(A) \neq \sigma(B)$, and σ maps distinct nonces from $M(a)$ into distinct nonces, distinct keys into distinct keys, and it has disjoint ranges for $M(a)$ and $Sub(t(a)) - M(a)$. σ is called *suitable for a sequence of actions* if it is suitable for each action in the sequence, and σ is called *suitable for a subset $C \subseteq \mathcal{T}_0$* if it is the identity on C.

An *event* of a protocol $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ is any triple $e_i = (u, \sigma, i)$, where $u = a_1 \cdots a_l$ is a role of $\mathcal{P}$, σ is a substitution suitable for u and $\mathcal{C}$, and $1 \leq i \leq l$. $\sigma(a_i)$ is the *action of the event e_i*. As usual, $act(e_i)$ ($t(e_i)$, $M(e_i)$) stands for the the action of e_i (term of e_i, set of new terms of e_i). The *local precedence relation* on events is defined

by $(u, \sigma, i) \rightarrow (u', \sigma', i')$ if and only if $u' = u$, $\sigma' = \sigma$, and $i' = i + 1$, provided that $i < |u|$. $\stackrel{+}{\rightarrow}$ is the transitive closure of $\rightarrow$. Given an event e, ${}^\bullet e$ stands for the *set of all local predecessors of* e, i.e., ${}^\bullet e = \{e' | e' \stackrel{+}{\rightarrow} e\}$.

Message generation rules. Given a set X of terms we denote by $analz(X)$ the least set of terms which includes X, contains t_1 and t_2 whenever it contains (t_1, t_2), and contains t whenever it contains $\{\{t\}_K\}_{K^{-1}}$ or $\{t\}_K$ and K^{-1}. By $synth(X)$ we denote the least set of terms which includes X, contains (t_1, t_2), for any terms $t_1, t_2 \in synth(X)$, and contains $\{t\}_K$, for any term t and key K in $synth(X)$. Moreover, $\overline{X}$ stands for $synth(analz(X))$.

States and runs. A *state* of a protocol $\mathcal{P}$ is an indexed set $s = (s_A | A \in \mathcal{A})$, where $s_A \subseteq \mathcal{T}$, for any agent A. The *initial state* is $s_0 = (s_{0A} | A \in \mathcal{A})$, where $s_{0A} = \mathcal{A} \cup \mathcal{C} \cup \mathcal{K}_A \cup Secret_A$, for any honest agent A, and $s_{0I} = \mathcal{A} \cup \mathcal{C} \cup \mathcal{K}_I \cup \mathcal{N}_I \cup \mathcal{K}_{0,I}$.

Given two states s and s' and an action a, we write $s[a\rangle s'$ if and only if:

1. if a is of the form $A!B : (M)t$, then:
   (a) (*enabling condition*) $t \in \overline{s_A \cup M}$ and $M \cap Sub(s) = \emptyset$;
   (b) $s'_A = s_A \cup M \cup \{t\}$, $s'_I = s_I \cup \{t\}$, and $s'_C = s_C$ for any $C \in \mathcal{A} - \{A, I\}$;
2. if a is of the form $A?B : t$, then:
   (a) (*enabling condition*) $t \in \overline{s_I}$;
   (b) $s'_A = s_A \cup \{t\}$ and $s'_C = s_C$, for all $C \in \mathcal{A} - \{A\}$.

We extend the notation "$[\cdot\rangle$" to events by letting $s[e\rangle s'$ whenever $s[act(e)\rangle s'$, and we call $s[e\rangle s'$ a *computation step*. The constraint "$M \cap Sub(s) = \emptyset$" is called the *freshness check condition*. It requires that the nonces and short-term keys generated by A in order to compose t (i.e., those in the set M) be fresh.

There is another natural way to define computation steps in a security protocol, namely by removing the freshness check condition. More precisely, the enabling condition embraces the form "$t \in \overline{s_A \cup M}$", all the other elements of the definition of a computation step remaining unchanged.

To distinguish between the two cases when discussing about security protocols and their computation steps we will use the terminology "*security protocol with freshness check*", for the first case, and "*security protocol without freshness check*", for the second one. When no distinction is necessary we will simply say "*security protocol*".

A *computation* or *run* of a security protocol is any sequence of computation steps

$$s_0[e_1\rangle s_1[\cdots[e_k\rangle s_k,$$

also written as $s_0[e_1 \cdots e_k\rangle s$ or even $e_1 \cdots e_k$, such that $s_{i-1}[e_i\rangle s_i$ for any $1 \leq i \leq k$, and ${}^\bullet e_i \subseteq \{e_1, \ldots, e_{i-1}\}$ for any $1 \leq i \leq k$ (for $i = 1$, ${}^\bullet e_i$ should be empty).

The secrecy problem. We say that a term $t \in \mathcal{T}_0$ is *secret at a state* s if $t \in analz(s_A) - analz(s_I)$, for some honest agent A. t is *secret along a run* ξ if it is secret at s, where $s_0[\xi\rangle s$. A run ξ is called *leaky* if there exists $t \in \mathcal{T}_0$ and a proper prefix ξ' of ξ such that t is secret along ξ' but not along ξ. The *secrecy problem* for security protocols is the problem to decide whether a security protocol has leaky runs.

If we replace the set $\mathcal{T}_0$ by $\bigcup_{A \in Ho} Secret_A$ in all the definitions above, we obtain a particular case of the secrecy problem, called the *initial secrecy problem*. That is, the initial secrecy problem is the problem to decide whether a given security protocol has runs ξ such that $t \in analz(s_I)$ for some initial secret t, where $s_0[\xi\rangle s$.

2 Complexity of Secrecy for Bounded Protocols

Let $\mathcal{P}$ be a protocol, $T \subseteq \mathcal{T}_0$ a finite set, and $k \geq 1$. A run of $\mathcal{P}$ is called a (T, k)-*run* if all terms in the run are built up upon T and all messages communicated in the course of the run have length at most k. When for $\mathcal{P}$ only (T, k)-runs are considered we will say that it is a *protocol under* (T, k)-*runs* or a (T, k)-*bounded protocol*, and denote this by $(\mathcal{P}, T, k)$. A *bounded protocol* is a (T, k)-bounded protocol, for some finite set $T \subseteq \mathcal{T}_0$ and $k \geq 1$.

The (*initial*) *secrecy problem* for (T, k)-bounded protocols is formulated with respect to (T, k)-runs only, by taking into consideration the set T instead of $\mathcal{T}_0$.

Let $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ be a (T, k)-bounded protocol. Then:

1. the number of messages communicated in the course of any (T, k)-run of $\mathcal{P}$ is bounded by
$$k^3 |T|^{\frac{k+1}{2}} = 2^{3 \log k + \frac{k+1}{2} \log |T|};$$

2. the number of instantiations (substitutions) of a given role u of $\mathcal{P}$ with messages of length at most k over T is bounded by
$$\left(2^{3 \log k + \frac{k+1}{2} \log |T|}\right)^{|u|(\frac{k+1}{2}+2)}$$
(u has exactly $|u|$ actions, and each action has at most $\frac{k+1}{2} + 2$ elements that can be substituted);

3. the number of (T,k)-*events* of $\mathcal{P}$ (i.e., events that can occur in all (T,k)-runs of $\mathcal{P}$) is bounded by

$$\begin{array}{rcl} \text{number of } (T,k)\text{-events} & \leq & \sum_{u \in role(\mathcal{P})} |u| \cdot 2^{(3\log k + \frac{k+1}{2}\log|T|)|u|(\frac{k+1}{2}+2)} \\ & \leq & \sum_{u \in role(\mathcal{P})} |u| \cdot 2^{(3\log k + \frac{k+1}{2}\log|T|)|w|(\frac{k+1}{2}+2)} \\ & = & |w| \cdot 2^{(3\log k + \frac{k+1}{2}\log|T|)|w|(\frac{k+1}{2}+2)} \\ & = & 2^{\log|w| + (3\log k + \frac{k+1}{2}\log|T|)|w|(\frac{k+1}{2}+2)} \end{array}$$

   where $role(\mathcal{P})$ is the set of all roles of $\mathcal{P}$.

Define the *size* of a (T,k)-bounded protocol $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ as being

$$size(\mathcal{P}) = |w| + k\log|T|.$$

This is a realistic measure. It takes into consideration the number of actions and the maximum number of bits necessary to represent messages of length at most k (we remark that the size of the representation of $\mathcal{P}$ is polynomial in $size(\mathcal{P})$). As we can see, the number of events that can occur in all (T,k)-runs of a (T,k)-bounded protocol is exponential in $poly(size(\mathcal{P}))$, for some polynomial $poly$.

In [4], several DEXPTIME-completeness results for the initial secrecy problem for bounded protocols in the restricted form and with bounded existentials have been proposed (the concept of "existential" in [4] corresponds to the concept of "freshness check" in our paper. In Section 2.3 we will provide the reader with full details about the relationship between the formalism in [4] and the one used in this paper). Unfortunately, all these results are wrong just because they are based on an algorithm which does not work for the case of protocols with existentials (see Section 2.3). In fact, it turns out that the presence of existentials requires much more effort than their absence, making the secrecy problem complete for NEXPTIME. We will prove this result as follows. First, by means of a slight variation of an algorithm in [4] adapted to the formalism used in this paper we show that the (initial) secrecy problem for bounded security protocols without freshness check is DEXPTIME-complete (Section 2.1). Then, we point out why this algorithm does not work for the case of freshness check and we show that in this case the (initial) secrecy problem is NEXPTIME-complete (Section 2.2). The last section recalls the MSR formalism [4], points out why the DEXPTIME-completeness results in [4] are wrong, and then corrects them. We also provide an undecidability result related to a problem left open in [4] (Table 9, page 282).

2.1 The Initial Secrecy Problem for Bounded Protocols without Freshness Check

Recall that the protocols without freshness check are obtained by removing the requirement "$M \cap Sub(s) = \emptyset$" from the enabling condition (see Section 1). The algorithm A1 below, which is a slight variation of the algorithm in [4] adapted to the formalism used in this paper, decides whether or not a bounded protocol without freshness check has leaky runs with respect to initial secrets.

Algorithm A1

`input:` bounded protocol $(\mathcal{P}, T, k)$ without freshness check;
`output:` "leaky protocol" if $\mathcal{P}$ has some leaky (T,k)-run w.r.t. initial secrets, and "non-leaky protocol", otherwise;

`begin`
1. `let` E' be the set of all (T,k)-events of $\mathcal{P}$;
2. $\xi := \lambda$; $s := s_0$; % λ denotes the empty sequence
3. `repeat`
4. $E := E'$;
5. $E' := \emptyset$;
6. $bool := 0$;
7. `while` $E \neq \emptyset$ `do`
8. `begin`
9. `choose` $e \in E$;
10. $E := E - \{e\}$;
11. `if` $s_0[\xi\rangle s[e\rangle s'$ `then`
12. `begin`
13. $s := s'$; $\xi := \xi e$; $bool := 1$;
14. `end`
15. `else` $E' := E' \cup \{e\}$;
16. `end`
17. `until` $bool = 0$;
18. `if` $(\bigcup_{A \in Ho} Secret_A) \cap analz(s_I) \neq \emptyset$
19. `then` "leaky protocol" `else` "non-leaky protocol"

`end.`

In the algorithm A1, E' is the set of all events that could not be applied in the previous cycle. Initially, E' is the set of all events of the protocol. The boolean variable *bool* takes the value 0 when no event in E can be applied.

The algorithm generates a run in which each event is applied at most once. The run is also maximal in the sense that it cannot be extended further by any new event (not already appearing in the run). Then, the algorithm checks whether this run is

leaky with respect to initial secrets. If the run is leaky, then clearly the protocol is leaky. Vice-versa, if the protocol is leaky with respect to initial secrets, then any leaky run can be extended to a maximal leaky run (in the sense discussed above). Moreover, this leaky run leads to exactly the same final state as the maximal run generated by the algorithm. This is due to the fact that in the absence of freshness check the following *persistence property* holds true: if an event is enabled at a state s, then it will be enabled at any state reachable from s (events are persistent once they become enabled). Therefore, the maximal run generated by the algorithm is leaky.

According to the discussion above, the order in which events are chosen to generate a maximal run (for bounded protocols without freshness check) is irrelevant. This is why in line 9 of the algorithm A1 we have used a generic statement "`choose` $e \in E$" without specifying any criterion for selecting events from E.

The algorithm terminates in exponential time with respect to the size of the protocol. To see that, let us count first the number of `while` cycles (in all `repeat` cycles). If no event in E can extend the current run ξ, then the algorithm terminates. Otherwise, all events in E that can extend the current run, taken in an arbitrary but fixed order, are applied in one `repeat` cycle. The next cycle will process, in the same way, the elements of E that could not be applied in the previous cycle. Therefore, the number of `repeat` cycles is bounded by $|E|$ (each cycle applies at least one event, except for the last one). The number of `while` cycles in the first `repeat` cycle is $|E|$, in the second `repeat` cycle is at most $|E| - 1$, and so on. Therefore, the number of `while` cycles in all `repeat` cycles is bounded by $|E| + (|E| - 1) + \cdots + 1 = \mathcal{O}(|E|^2)$.

Given s and ξ such that $s_0[\xi\rangle s$, the test "$s[e\rangle s'$" in line 11 of the algorithm can be performed in polynomial time with respect to the number of events, and this is true for the test in line 18 as well.

Therefore, the complexity of the algorithm is exponential in $poly(size(\mathcal{P}))$, for some polynomial $poly$, showing that the initial secrecy problem for bounded protocols without freshness check is in DEXPTIME.

It can be shown that the initial secrecy problem for bounded protocols without freshness check is DEXPTIME-hard by reducing the membership problem for unary logic programs [1] to this problem. Recall first the concept of a unary logic program. Let Σ be a set consisting of one constant symbol $\perp$ and finitely many unary function symbols, let $Pred$ be a finite set of unary predicate symbols, and x be a variable. A *unary logic program* over Σ, $Pred$, and x is a finite set of clauses of the form

$$p_0(t_0) \leftarrow p_1(t_1), \ldots, p_n(t_n)$$

or

$$p_0(t_0) \leftarrow true,$$

where $p_0, \ldots, p_n \in Pred$, and $t_0, \ldots, t_n$ are terms over $\Sigma \cup \{x\}$ with t_0 being flat, that is, $t_0 \in \{\bot, x, f(x) | f \in \Sigma - \{\bot\}\}$. Moreover, all clauses with $p_0(\bot)$ in the head have only *true* in the body.

An *atom* is a construct of the form $p(t)$, where $p \in Pred$ and t is a term. If t is a *ground term*, that is, it does not contain x, then $p(t)$ is called a *ground atom*. A *proof tree* for a ground atom $p(t)$ under a unary logic program LP is any tree that satisfies:

- its nodes are labeled by ground atoms;
- the root is labeled by $p(t)$;
- each intermediate node which is labeled by some B has children labeled by $B_1, \ldots, B_n$, where $B \leftarrow B_1, \ldots, B_n$ is a ground instance of a clause in LP (i.e., the variable x is substituted by ground terms over Σ);
- all the leaves are labeled by *true*.

The *membership problem for unary logic programs* is the problem to decide, given a logic program LP and a ground atom $p(t)$, whether there exists a proof tree for $p(t)$ under LP. In [1] it has been proved that this problem is DEXPTIME-complete (being equivalent to the type-checking problem for path-based approximation for unary logic programs).

Theorem 2.1 The initial secrecy problem for bounded protocols without freshness check is DEXPTIME-hard.

Proof Let LP be a unary logic program over some Σ, $Pred$, and x, and let $p(t)$ be a ground atom over Σ and $Pred$. Define a security protocol $\mathcal{P}$ as follows:

- to each element $e \in \Sigma \cup Pred \cup \{x\}$ associate a nonce u_e. Except for u_x, all these nonces are constants of the protocol;
- encode terms and atoms as follows:
  - $\langle e \rangle = u_e$, for all $e \in \{\bot, x\}$;
  - $\langle f(t) \rangle = (u_f, \langle t \rangle)$, for any unary function symbol f and term t;
  - $\langle p(t) \rangle = (u_p, \langle t \rangle)$, for any predicate symbol p and term t.
- consider the agents A_C, B_C, E and F, for any clause C. It is assumed that they are pairwise distinct;
- consider a key K known only by the honest agents;

- $Secret_F = \{y\}$, where y is a distinct nonce, and $Secret_X = \emptyset$, for all $X \neq F$;
- to each clause $C:\ p_0(t_0) \leftarrow p_1(t_1), \ldots, p_n(t_n)$ we associate the role

$$\begin{array}{lcl} A_C?B_C & : & \{\langle p_1(t_1)\rangle\}_K, \ldots, \{\langle p_n(t_n)\rangle\}_K \\ A_C!B_C & : & \{\langle p_0(t_0)\rangle\}_K \end{array}$$

- to each clause $C:\ p_0(t_0) \leftarrow true$ we associate the role

$$A_C!B_C \ : \ \{\langle p_0(t_0)\rangle\}_K$$

- the following role reveals a secret if $p(t)$ has a tree proof under LP:

$$\begin{array}{lcl} F?E & : & \{\langle p(t)\rangle\}_K \\ F!E & : & y \end{array}$$

We consider the protocol $\mathcal{P}$ under (T,k)-runs, where T is the set of all elements mentioned above (nonces, agents, the key K, and the nonce y) and k is a suitable chosen constant (the maximum length of some clause under some instantiation used to decide the membership of $p(t)$). Then, it is easily seen that $p(t)$ has a tree proof in LP if and only if the protocol $\mathcal{P}$ under (T,k)-runs reveals the secret. □

Corollary 2.1 The initial secrecy problem for bounded protocols without freshness check is DEXPTIME-complete.

2.2 The (Initial) Secrecy Problem for Bounded Protocols with Freshness Check

The algorithm A1 given in the previous section cannot be applied to the (initial) secrecy problem for bounded protocols with freshness check. We will illustrate this by considering two examples, one for the initial secrecy problem and one for the secrecy problem.

Example 2.1 Let $\mathcal{P}_1$ be the protocol given below:

$$\begin{array}{lcl} A!B & : & (\{K\})x_0, K \\ B?A & : & x_0, K \\ C!D & : & (\{K'\})K' \end{array}$$

In this protocol, A, B, C, and D are constants, x_0 is an initial secret of A, and K and K' are keys in a finite non-empty set of short-term keys.

Clearly, the protocol is leaky. However, if the algorithm A1 applies first all the events based on the third action, then no event based on the first action will be enabled. Therefore, the algorithm generates a non-leaky maximal run, and concludes that the protocol is not leaky.

It should be clear that the algorithm A1 does not work for the secrecy problem either. To be more convincing we will provide an example.

Example 2.2 Let $\mathcal{P}_2$ be the protocol given below:

$$\begin{array}{lcl} A!B & : & (\{x\})\{x\}_K \\ B?A & : & \{x\}_K \\ A!B & : & K \\ C!D & : & (\{y\})y \end{array}$$

where A, B, C, and D are constants in the protocol, x and y are nonces, and K is a key.

The protocol is leaky. However, if all events based on the fourth action are applied first, then the algorithm A1 generates a non-leaky maximal run and concludes that the protocol is not leaky.

All these examples show that the order in which events are applied when freshness check is required, is crucial. This makes the secrecy problem for the case "with freshness check" considerable harder than the case "without freshness check".

Theorem 2.2 The (initial) secrecy problem for bounded protocols with freshness check is NEXPTIME-complete.

Proof The following non-deterministic algorithm decides the secrecy problem for bounded protocols with freshness check.

Algorithm A2

`input:` bounded protocol $(\mathcal{P}, T, k)$ with freshness check;
`output:` "leaky protocol" if $\mathcal{P}$ has leaky (T, k)-runs;
`begin`
  `let` E be the set of all (T, k)-events of $\mathcal{P}$;
  `guess` a sequence $\xi := e_1 \cdots e_m$ of pairwise
      distinct events from E;
  `if` ξ is a run `then`
    `begin`
      `let` $s_0[e_1\rangle s_1[\cdots[e_m\rangle s_m$;
      $Secret := \bigcup_{A \in Ho} Secret_A$;
      `for` $i := 1$ `to` $m - 1$ `do`
          $Secret := Secret \cup ((\bigcup_{A \in Ho} analz(s_{iA})) - analz(s_{iI}))$;
      `if` $Secret \cap analz(s_{mI}) \neq \emptyset$ `then` "leaky protocol";
    `end`
`end.`

If the "`for`" statement is dropped in Algorithm A2, the new algorithm will decide the initial secrecy problem for bounded protocols with freshness check.

It is easy to see that the algorithm is correct and terminates in non-deterministic exponential time with respect to $poly(size(\mathcal{P}))$, for some polynomial $poly$. Therefore, the (initial) secrecy problem for bounded protocols with freshness check is in NEXPTIME.

To prove completeness, we shall reduce any language in NEXPTIME to the initial secrecy problem for bounded protocols with freshness check (this is also sufficient for the secrecy problem). So, suppose that L is a language decided by a non-deterministic Turing machine $TM = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ in time 2^n, where Q is the set of states, Σ is the input alphabet, Γ is the tape alphabet, q_0 is the initial state, $\Box$ is the blank symbol, F is the set of final states, and $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{-1, 1\}$ is the transition relation (-1 specifies a move to the left, while 1 specifies a move to the right). A configuration of TM will be written in the form (q, α, a, β), where q is the current state, a is the symbol scanned by the tape head, α is the string to the left of the tape head, and β is the string to the right of the tape head.

For each input w of TM we construct a bounded protocol $(\mathcal{P}_{TM,w}, T, k)$ with freshness check whose size is polynomial in $n = |w|$ and such that w is accepted by TM in time 2^n if and only if $\mathcal{P}_{TM,w}$ under (T, k)-runs is leaky. First, we may assume that all the elements of $Q \cup \Gamma$ are nonces. We also consider a distinguished nonce \$. All these nonces are constants of the protocol. A set of $2^n + ||w||$ distinct nonces, which are not constants of the protocol, is also considered ($||w|| = |w| + 1$, if $w \neq \lambda$, and $||w|| = 2$, otherwise). The agents are A, B, $A_{t,b}$, $B_{t,b}$, $A_{t\$,b}$, $B_{t\$,b}$, $A_{b,t}$, $B_{b,t}$, $A_{b,t\$}$, $B_{b,t\$}$, $A_{\lambda,t}$, $B_{\lambda,t}$, $A_{q,b}$, $B_{q,b}$, $A_{q,b\$}$, and $B_{q,b\$}$, for any transition t of TM, $b \in \Gamma$, and final state q. All these honest agents will share a long-term key K. It is assumed that $Secret_{A_{q,b}} = Secret_{B_{q,b}} = Secret_{A_{q,b\$}} = Secret_{B_{q,b\$}} = \{x_0\}$ for any final state q and $b \in \Gamma$, where x_0 is a distinguished nonce, and $Secret_X = \emptyset$ for all the other agents X.

A configuration $(q, a_1 \cdots a_n, a, b_1 \cdots b_m)$ of TM is represented in the protocol $\mathcal{P}_{TM,w}$ by a sequence of the form

$$\{u_1, a_1, u_2\}_K, \ldots, \{u_n, a_n, u_{n+1}\}_K, \{u_{n+1}, (q, a), u_{n+2}\}_K,$$
$$\{u_{n+2}, b_1, u_{n+3}\}_K, \ldots, \{u_{n+m+1}, (b_m, \$), u_{n+m+2}\}_K,$$

where $u_1, \ldots, u_{n+m+2}$ are distinct nonces.

The protocol actions are:

- A initiates the protocol and sends $w = a_1 \cdots a_n$ to B:

$$\begin{cases} A!B \;:\; (\{u_1, u_2\})\{u_1, (q_0, (\Box, \$)), u_2\}_K, & \text{if } n = 0 \\ A!B \;:\; (\{u_1, u_2\})\{u_1, (q_0, (a_1, \$)), u_2\}_K, & \text{if } n = 1 \\ A!B \;:\; (\{u_1, \ldots, u_{n+1}\})\{u_1, (q_0, a_1), u_2\}_K, \{u_2, a_2, u_3\}_K, \ldots, & \\ \qquad\qquad \{u_{n-1}, a_{n-1}, u_n\}_K, \{u_n, (a_n, \$), u_{n+1}\}_K, & \text{if } n > 1; \end{cases}$$

- a transition $t = (q, a, q', a', -1)$ is simulated by:

$$\begin{array}{rcl} A_{t,b}?B_{t,b} & : & \{u, b, v\}_K, \{v, (q, a), z\}_K \\ A_{t,b}!B_{t,b} & : & (\{v'\})\{u, (q', b), v'\}_K, \{v', a', z\}_K \end{array}$$

or

$$\begin{array}{rcl} A_{t\$,b}?B_{t\$,b} & : & \{u, b, v\}_K, \{v, (q, (a, \$)), z\}_K \\ A_{t\$,b}!B_{t\$,b} & : & (\{v'\})\{u, (q', b), v'\}_K, \{v', (a', \$), z\}_K \end{array}$$

for any $b \in \Gamma$;

- a transition $t = (q, a, q', a', 1)$ is simulated by

$$\begin{array}{rcl} A_{b,t}?B_{b,t} & : & \{u, (q, a), v\}_K, \{v, b, z\}_K \\ A_{b,t}!B_{b,t} & : & (\{v'\})\{u, a', v'\}_K, \{v', (q', b), z\}_K \end{array}$$

or

$$\begin{array}{rcl} A_{b,t\$}?B_{b,t\$} & : & \{u, (q, a), v\}_K, \{v, (b, \$), z\}_K \\ A_{b,t\$}!B_{b,t\$} & : & (\{v'\})\{u, a', v'\}_K, \{v', (q', (b, \$)), z\}_K \end{array}$$

or

$$\begin{array}{rcl} A_{\lambda,t}?B_{\lambda,t} & : & \{u, (q, (a, \$)), v\}_K \\ A_{\lambda,t}!B_{\lambda,t} & : & (\{v'\})\{u, a', v'\}_K, \{v', (q', (\square, \$)), v\}_K \end{array}$$

for any $b \in \Gamma$;

- when a final configuration is reached, the secret is revealed:

$$\begin{array}{rcl} A_{q,b}?B_{q,b} & : & \{u, (q, b), v\}_K \\ A_{q,b}!B_{q,b} & : & x_0 \end{array}$$

or

$$\begin{array}{rcl} A_{q,b\$}?B_{q,b\$} & : & \{u, (q, (b, \$)), v\}_K \\ A_{q,b\$}!B_{q,b\$} & : & x_0 \end{array}$$

for any $q \in F$ and $b \in \Gamma$.

The size of the protocol $\mathcal{P}_{TM,w}$ is polynomial in n. The protocol depends by TM, but TM is constant for any instance w of L. The set of nonces can be considered as an initial fragment of the set of natural numbers and, therefore, it can be specified by giving just a natural number (the cardinality of this set). As a conclusion, the protocol can be constructed in polynomial time with respect to $n = |w|$.

Let T be the set of all basic terms of the protocol. It is clear that there exists $k \in \mathcal{O}(n)$ such that any computation of TM with at most 2^n steps can be simulated by a (T,k)-run of $\mathcal{P}_{TM,w}$. Moreover, if TM accepts w, then $\mathcal{P}_{TM,w}$ reveals the secret. Conversely, if a (T,k)-run ξ of the protocol reveals the secret, then there exists a computation of TM which accepts w. ξ cannot have more than 2^n+1 send events which generate new nonces because each send event generates at least one new nonce and the set of all non-constant nonces has cardinality $2^n + ||w||$ ($||w||$ distinct nonces are used to represent w). Therefore, the corresponding computation of TM cannot have more than 2^n steps.

This shows that L is reducible to the initial secrecy problem for bounded protocols with freshness check. □

Remark 2.1 In the proof of Theorem 2.2, all the roles associated to transitions change the linking nonce v into a fresh one v'. This is crucial for a correct simulation of the Turing machine; otherwise, it might happen that the intruder sends two linked terms $\{u,(q,a),v\}_K, \{v,b,z\}_K$, where $\{u,(q,a),v\}_K$ represents the content of a tape cell at a given step, while $\{v,b,z\}_K$ represents the content of an adjacent cell at a different step. In this way, different runs can be mixed leading to a leaky run with no corresponding computation in the Turing machine.

2.3 Relationship with the MSR Formalism

The algorithm A1 in Section 2.1, proposed initialy in [4] under the *multiset rewriting* (MSR) formalism for modeling security protocols, was used to develop several complexity results for the secrecy problem for bounded protocols "with existentials" [4]. Unfortunately, all the results based on this algorithm are wrong. In order to show that we recall first the MSR formalism. This formalism is based on:

- a *signature*, which specifies a set of *sorts* (for keys, messages, nonces etc.) together with function and predicate symbols (each symbol having associated a specific type). Function symbols with no arguments are also called *constant symbols*;
- a set of *variables*, each of which having associated a sort;
- *terms*, which are defined as usual;
- *atomic formulas*, which are constructs of the form $P(t_1,\ldots,t_n)$, where P is a predicate symbol of type $s_1\cdots s_n$ and t_i is a term of sort s_i, for any i;
- *facts*, which are atomic formulas $P(t_1,\ldots,t_n)$, where all terms t_i are *ground terms* (i.e., variable-free terms);

- *rules*, which are constructs of the form

$$F_1, \ldots, F_k \ \rightarrow \ \exists x_1, \ldots, \exists x_l.G_1, \ldots, G_p,$$

  where the F's and G's are atomic formulas and the x's are variables;

- *states*, which are multisets of facts.

If S is a state,

$$r: \quad F_1, \ldots, F_k \ \rightarrow \ \exists x_1, \ldots, \exists x_l.G_1, \ldots, G_p$$

is a rule, and σ is a ground substitution such that $\sigma(x_i)$ is a new constant symbol (that is, not previously generated) and $\sigma(F_j) \in S$, for all i and j, then the rule r can be applied to S yielding a new state S'. This state is obtained from S by

$$S' = (S - \langle \sigma(F_j) | 1 \leq j \leq k \rangle) \cup \langle \sigma(G_j) | 1 \leq j \leq p \rangle,$$

where "$\langle \cdots \rangle$" denotes multisets and the difference and union with multisets are defined as usual. We denote this by $S \stackrel{r,\sigma}{\Longrightarrow} S'$.

Existential quantifiers in the right hand side of rules, simply called *existentials*, capture the idea of "generation of new elements" (keys, nonces etc.). For example, the new elements generated in the computation step $S \stackrel{r,\sigma}{\Longrightarrow} S'$ are $\sigma(x_1), \ldots, \sigma(x_l)$.

A *MSR theory* consists of a signature and a set of rules over this signature. In order to suitably model security protocols and to gain strong decidability and complexity results on the security protocols, MSR theories were subjected to two main constraints in [4]: well-foundedness and restrictedness. To explain these we need a few concepts.

Let $\mathcal{T}$ be a MSR theory and P be a predicate of arity n. A rule $l \rightarrow r$ in $\mathcal{T}$ *creates* P *facts* if some $P(t_1, \ldots, t_n)$ occurs more times in r than in l. $l \rightarrow r$ *consumes* P *facts* if some $P(t_1, \ldots, t_n)$ occurs more times in l than in r. $l \rightarrow r$ *preserves* P *facts* if every $P(t_1, \ldots, t_n)$ occurs the same number of times in r as in l. The predicate P is *persistent* in $\mathcal{T}$ if every rule in $\mathcal{T}$ which contains P either creates or preserves P facts.

A MSR theory $\mathcal{T}$ is called a *well-founded protocol theory* if $\mathcal{T}$ is a disjoint union of MSR theories $\mathcal{T} = \mathcal{I} \uplus \mathcal{R} \uplus \mathcal{A}_1 \uplus \cdots \uplus \mathcal{A}_n$, where:

- for any $1 \leq i \leq n$, there exists a finite list of predicates $A^i_0, \ldots, A^i_{k_i}$, called *role states*, such that:
  - for any rule $l \rightarrow r \in \mathcal{A}_i$ there exists exactly one A^i_p in l and exactly one A^i_q in r, and $p < q$;

– no role state predicate that occurs in rules in $\mathcal{A}_i$ can occur in rules in $\mathcal{A}_j$, for any $i \neq j$.

$\mathcal{A}_i$ is usually called a *bounded role theory* and A_0^i is the *initial role state* of this theory;

- $\mathcal{I}$, called the *initialization theory*, is a set of rules such that all formulas in the right hand side of rules in $\mathcal{I}$ either contains existentials or are persistent in $\mathcal{T}$. Moreover, rules in $\mathcal{I}$ do not contain role state predicates;

- $\mathcal{R}$, called the *role generation theory*, is a set of rules of the form

$$P(s_1, \ldots, s_l), Q(t_1, \ldots, t_m), \ldots \rightarrow$$

$$A_0(r_1, \ldots, r_p), P(s_1, \ldots, s_l), Q(t_1, \ldots, t_m), \ldots$$

where $P(s_1, \ldots, s_l), Q(t_1, \ldots, t_m), \ldots$ is a finite list of persistent facts created by $\mathcal{I}$ and not involving any role state, and A_0 is the initial role state of some $\mathcal{A}_1, \ldots, \mathcal{A}_n$;

- $\mathcal{I}$ precedes $\mathcal{R}$ and $\mathcal{R}$ precedes $\mathcal{A}_1, \ldots, \mathcal{A}_n$ (a theory $\mathcal{T}_1$ precedes a theory $\mathcal{T}_2$ if no fact that appear in the left hand side of some rule in $\mathcal{T}_1$ is created by some rule in $\mathcal{T}_2$).

The *restricted form* of MSR theories is obtained by imposing two more constraints on top of well-foundedness:

- there are two finite lists $N_{R1}, \ldots, N_{Rm}$ of network predicates for receive actions and $N_{S1}, \ldots, N_{Sm}$ of network predicates for send actions such that, for any role i, its rules are of the form

$$A_p^i(\ldots), N_{Rj}(\ldots) \rightarrow \exists \ldots A_q^i(\ldots) N_{Sl}(\ldots)$$

where $p < q$ and $j < l$;

- the initialization theory is a set of ground facts.

In [4], only security protocols modeled as well-founded protocol theories are studied.

The MSR formalism has been extended in [4] with *tests for disequality*, which are constructs of the form "$t_1 \neq t_2$", where t_1 and t_2 are terms. These tests may be added only to the left hand side of rules. For example,

$$P(t_1), Q(t_2), t_1 \neq t_2 \ \rightarrow \ \exists x.R(x)$$

is such a rule. It can be applied to a state S under a substitution σ if the state S contains two facts $\sigma(P(t_1))$ and $\sigma(Q(t_2))$ such that $\sigma(t_1) \neq \sigma(t_2)$.

The mechanism based on existentials in the MSR formalism is equivalent to the freshness check mechanism used with the formalism in this paper. Disequality tests somehow add more power to the MSR formalism and they do not have any equivalent in the formalism in this paper. An "intruder with no existentials" in the MSR formalism means that the intruder does not generate new elements and, therefore, it is equivalent to an *intruder without generation* (i.e., $\mathcal{N}_I \cup \mathcal{K}_{0,I} = \emptyset$) in this paper. But an "intruder with existentials" in the MSR formalism, which means that the intruder can generate new elements from an arbitrary (possible infinite) set of elements does not have any equivalent in the formalism used in this paper because there is no freshness check mechanism for the intruder in the formalism in this paper. An "intruder with bounded existentials" in the MSR formalism means that the intruder can generate new elements from a finite set of elements; this concept still does not have any equivalent in the formalism used in this paper for the same reason as that above (for a complete formalization of the intruder in the MSR formalism the reader is referred to [4], page 264). A "protocol with bounded existentials" in [4] means that the honest agents can generate new elements from a given finite set.

As a conclusion of our discussion above we can say that any protocol in the formalism in this paper for which the intruder is without generation can be easily translated into the MSR formalism:

- if the protocol is without freshness check then the corresponding protocol in the MSR formalism does not have existentials and disequality tests, and the intruder is with no existentials;
- if the protocol is with freshness check then the corresponding protocol in the MSR formalism may have existentials but it does not have disequality tests and the intruder is with no existentials

(we emphasize that the translation does not assure either restrictedness or well-foundedness).

Example 2.3 The protocol in Example 2.1 can be translated into the MSR formalism as shown below:

$$
\begin{array}{lcl}
 & \rightarrow & A_0(x_0) \\
 & \rightarrow & B_0() \\
 & \rightarrow & C_0() \\
A_0(x_0), N_{R1}() & \rightarrow & \exists K.A_1(x_0, K), N_{S2}(x_0, K) \\
B_0(), N_{R2}(x_0, K) & \rightarrow & B_1(x_0, K), N_{S3}() \\
C_0(), N_{R3}() & \rightarrow & \exists K.C_1(K), N_{S4}(K)
\end{array}
$$

In this protocol, A_0, B_0, and C_0 are initial role states, and A_1, B_1, and C_1 are role states. The first three rules define the role generation theory; they initializes the roles for A, B, and C, allowing an unlimited number of sessions to be started for any principal acting in these roles.

The fourth rule says that A, knowing the secret x_0 in its initial state, generates a key K, moves to the new state A_1 where both x_0 and K are memorized, and sends the message "x_0, K" on the network ($N_{S2}(x_0, K)$). The fifth rule says that B, being in its initial state and receiving "x_0, K", moves to a new state B_1 where both x_0 and K are memorized. The sixth rule is interpreted in a similar way to fourth rule.

One can easily see that the protocol above is in restricted form.

Now, we obtain the following results (in the MSR formalism).

Corollary 2.2 The following four problems, in the MSR formalism, are NEXPTIME-complete:

1. the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials;

2. the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, with disequality tests, and an intruder with no existentials;

3. the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials.

4. the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, with disequality tests, and an intruder with bounded existentials.

Proof First, we transform the third problem into an equivalent problem by replacing the "intruder with existentials" by an intruder with no existentials and which has a distinguished nonce n and a distinguished short-term key K in the initial state (as in [4], the last paragraph of section 5.4.2). This transformation is simply performed by removing intruder's rule for generating new data and adding the ground facts $M(n)$ and $M(K)$ to the initial state. These ground facts say that the intruder knows initially the nonce n and the key K, and can use them during any computation. The protocol such obtained is equivalent to the original one (which is assumed to be a protocol in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials) in the sense that

it does not change the status of the secrecy problem of the original protocol. This is simply obtained just because of the absence of disequality tests.

With the transformation above, every instance of any problem in the corollary defines a finite set of ground facts and ground instances of the rules. Now, to show that all these problems are in NEXPTIME, one can easily develop a non-deterministic version of the algorithm in [4] (page 284). This algorithm works similarly to our algorithm A2: it guesses a order in which all ground instances of the rules can be applied, and then applies the rules in this order (as long as they can be applied). At the end, the algorithm checks whether the intruder gets any initial secrets.

In order to show that these problems are NEXPTIME-complete it is sufficient to show that they include a NEXPTIME-complete subproblem. In the proof of Theorem 2.2 we may consider that the protocol satisfies $\mathcal{N}_I \cup \mathcal{K}_{0,I} = \emptyset$ (i.e., the intruder is without generation). As a conclusion, the initial secrecy problem for bounded protocols with freshness check and an intruder without generation is NEXPTIME-complete as well. Now, it is sufficient to show that any protocol in the proof of Theorem 2.2, which simulates non-deterministic Turing machines, can be equivalently translated (from the initial secrecy problem point of view) into a protocol in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials. First, we may assume that all the elements in $Q \cup \Gamma$ are constants of sort nonce. We consider a distinguished constant of sort nonce, \$, a long-term key K and a secret information x_0 known only to the honest agents, the variables u, b, v, v', z of sort nonce and the predicates A_0^t, A_1^t, B_0^t, B_1^t, C_0^t, C_1^t, A_0^q, A_1^q, B_0^q, B_1^q, N_{Ri}, and N_{Si}, for any $i = 1, 2, 3, 4$, transition t, and final state q.

The current configuration of the Turing machine is encoded as in the proof of Theorem 2.2, and the initial state of the new protocol (in the MSR formalism), for an input $w = a_1 \cdots a_n$ of the machine, consists of the following ground facts:

- $N_{S1}(\{u_1, (q_0, (\Box, \$)), u_2\}_K)$, if $n = 0$;
- $N_{S1}(\{u_1, (q_0, (a_1, \$)), u_2\}_K)$, if $n = 1$;
- $N_{S1}(\{u_1, (q_0, a_1), u_2\}_K)$, $N_{S1}(\{u_2, a_2, u_3\}_K)$, $\ldots$, $N_{S1}(\{u_{n-1}, a_{n-1}, u_n\}_K)$, if $n > 1$.

The rules of the new protocol are:

- the role generation theory consists of all rules $\rightarrow A_0^t()$, $\rightarrow B_0^t()$, $\rightarrow C_0^t()$, $\rightarrow A_0^q()$, and $\rightarrow B_0^q()$, for any transition t and final state q;
- for each transition $t = (q, a, q', a', -1)$ two role theories are included, each of which consists of exactly one rule:

$$A_0^t(), N_{R1}(\{u, b, v\}_K, \{v, (q, a), z\}_K) \rightarrow$$

$$\exists v'.N_{S2}(\{u,(q',b),v'\}_K,\{v',a',z\}_K),A_1^t()$$

and

$$B_0^t(),N_{R1}(\{u,b,v\}_K,\{v,(q,(a,\$)),z\}_K)\rightarrow$$
$$\exists v'.N_{S2}(\{u,(q',b),v'\}_K,\{v',(a',\$),z\}_K),B_1^t()$$

- for each transition $t=(q,a,q',a',1)$ three role theories are included, each of which consists of exactly one rule:

$$A_0^t(),N_{R1}(\{u,(q,a),v\}_K,\{v,b,z\}_K)\rightarrow$$
$$\exists v'.N_{S2}(\{u,a',v'\}_K,\{v',(q',b),z\}_K),A_1^t()$$

and

$$B_0^t(),N_{R1}(\{u,(q,a),v\}_K,\{v,(b,\$),z\}_K)\rightarrow$$
$$\exists v'.N_{S2}(\{u,a',v'\}_K,\{v',(q',(b,\$)),z\}_K),B_1^t()$$

and

$$C_0^t(),N_{R2}(\{u,(q,(a,\$)),v\}_K)\rightarrow$$
$$\exists v'.N_{S3}(\{u,a',v'\}_K,\{v',(q',(\square,\$)),v\}_K),C_1^t()$$

- two role theories, each of which consists of exactly one rule,

$$A_0^q(),N_{R3}(\{u,(q,b),v\}_K)\rightarrow N_{S4}(x_0),A_1^q()$$

and

$$B_0^q(),N_{R3}(\{u,(q,(b,\$)),v\}_K)\rightarrow N_{S4}(x_0),B_1^q()$$

are also included for each final state q.

It is easy to see that this protocol is in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials; moreover, it does exactly the same job as the protocol in the proof of Theorem 2.2. Therefore, the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials includes a NEXPTIME-complete problem, which shows that it is NEXPTIME-complete too.

As the second, the third, and the fourth problem includes the first problem as a subproblem, it follows that these problems are NEXPTIME-complete too. □

Corollary 2.2 corrects three false statements in [4] according to which the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, with or without disequality tests, and an intruder with no

existentials, and the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials, are DEXPTIME-complete. The source of these mistakes was that the authors in [4] claimed that the algorithm A1 works in the case of existentials because, as the authors said, "... bounding the number of protocol existentials means that we can generate all the existentials used by all runs of the protocol during the initialization phase". However, this is false as the protocol in Example 2.3 shows:

$$\begin{array}{lcl} & \rightarrow & A_0(x_0) \\ & \rightarrow & B_0() \\ & \rightarrow & C_0() \\ A_0(x_0), N_{R1}() & \rightarrow & \exists K.A_1(x_0, K), N_{S2}(x_0, K) \\ B_0(), N_{R2}(x_0, K) & \rightarrow & B_1(x_0, K), N_{S3}() \\ C_0(), N_{R3}() & \rightarrow & \exists K.C_1(K), N_{S4}(K) \end{array}$$

By the role generation theory (the first three rules) the ground facts $A_0(x_0)$, $B_0()$, and $C_0()$ can be "pumped" unboundedly. Therefore, if all ground instances of the sixth rule are applied first, the algorithm A1 generates a non-leaky maximal run and concludes that the protocol is not leaky, while the protocol is leaky.

We would like to point out that the status of the initial secrecy problem for protocols in restricted form, with bounded length messages, no existentials, and an intruder with no existentials was correctly identified in [3] as being DEXPTIME-complete.

In [4], the status of the initial secrecy problem for protocols in restricted form, with bounded length messages, bounded existentials, disequality tests and an intruder with existential was left open. We do not have any solution to this problem but we can prove that the problem is undecidable if the protocols are not well-founded (but satisfy the other requirements listed above).

Theorem 2.3 The initial secrecy problem for non-well-founded MSR protocols, with bounded length messages, no existentials, disequality tests and an intruder with existentials is undecidable.

Proof We will reduce the halting problem for deterministic Turing machines to this problem. Given a deterministic Turing machine and a input for this machine we will construct a non-well-founded protocol with bounded length messages, no existentials, disequality tests and an intruder with existentials such that the Turing machine halts on the input iff the protocol is leaky. The construction follows the same idea as in the proof of Theorem 2.2 adapted to the MSR formalism (and except for the fact that in this case the Turing machines are deterministic).

Let $TM = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ be a deterministic Turing machine (Q, Σ, Γ, q_0, $\Box$, and F are as in the proof of Theorem 2.2, and $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{-1, 1\}$ defines a partial function from $Q \times \Gamma$ into $Q \times \Gamma \times \{-1, 1\}$), and let w be an input for TM. We construct a non-well-founded protocol $\mathcal{P}_{TM,w}$ with bounded length messages, no existentials, disequality tests and an intruder with existentials such that TM halts on w if and only if $\mathcal{P}_{TM,w}$ is leaky. First, we may assume that all the elements in $Q \cup \Gamma$ are constants of sort nonce. We consider three distinguished constants of sort nonce, \$, *init*, and $\perp$, a long-term key K and a secret information x_0 known only to the honest agents, the variables x, y, u, b, v, v', z of sort nonce and the predicates A_0^t, A_1^t, B_0^t, B_1^t, C_0^t, C_1^t, A_0^q, A_1^q, B_0^q, B_1^q, $T_{q',a}$, $T_{q',a\$}$, L, N_{Ri}, and N_{Si}, for any $i = 1, 2, 3, 4$, transition $t \in \delta$, final state q, state q', and tape symbol a. Type A (type B, type C, type N) predicates will be used as in the proof of Corollary 2.2. L is used to define a list of distinct nonces. The first element of the list is *init* and the last element is $\perp$. $L(x, y)$ means that x and y are linked in the list. $T_{q',a}(x, y, u, b, z)$ and $T_{q',a\$}(x, y, u, b, z)$ means that the nonce x, which is an element of the list, is to be compared against the nonce y and the nonces u, b, z will be used to compute the new configuration of the machine. These predicates will be used in conjunction with the predicate L and disequality tests: when a nonce is received, it is compared with all nonces in the list; if it is different than all nonces in the list then it is appended to the list. Only nonces in this list will be used to simulate the computation of the Turing machine starting with w.

The current configuration of the Turing machine is encoded as in the proof of Theorem 2.2, and the initial state of the new protocol (in the MSR formalism), for an input $w = a_1 \cdots a_n$ of the machine, consists of the following ground facts:

- $N_{S1}(\{u_1, (q_0, (\Box, \$)), u_2\}_K)$, $L(init, u_1)$, $L(u_1, u_2)$, $L(u_2, \perp)$, if $n = 0$;
- $N_{S1}(\{u_1, (q_0, (a_1, \$)), u_2\}_K)$, $L(init, u_1)$, $L(u_1, u_2)$, $L(u_2, \perp)$, if $n = 1$;
- $N_{S1}(\{u_1, (q_0, a_1), u_2\}_K)$, $N_{S1}(\{u_2, a_2, u_3\}_K)$, $\ldots$, $N_{S1}(\{u_{n-1}, a_{n-1}, u_n\}_K)$, $N_{S1}(\{u_n, (a_n, \$), u_{n+1}\}_K)$, $L(init, u_1)$, $\ldots$, $L(u_n, u_{n+1})$, $L(u_{n+1}, \perp)$, if $n > 1$.

The protocol rules are:

- the role generation theory consists of the rules $\rightarrow A_0^t()$, $\rightarrow B_0^t()$, $\rightarrow C_0^t()$, $\rightarrow A_0^q()$, and $\rightarrow B_0^q()$, for any transition t and final state q;
- for each transition $t = (q, a, q', a', -1)$ two role theories are included, each of which consists of three rules:

$$\begin{array}{lll} A_0^t(), N_{R1}(\{u, b, v\}_K, \{v, (q, a), z\}_K, v') & \rightarrow & T_{q,a}(init, v', u, b, z) \\ T_{q,a}(x, v', u, b, z), L(x, y), x \neq v', y \neq \perp & \rightarrow & L(x, y), T_{q,a}(y, v', u, b, z) \end{array}$$

$$T_{q,a}(x,v',u,b,z), L(x,\bot), x \neq v', \bot \neq v' \rightarrow \\ L(x,v'), L(v',\bot), N_{S2}(\{u,(q',b),v'\}_K, \{v',a',z\}_K), A_1^t()$$

and

$$\begin{array}{ll} B_0^t(), N_{R1}(\{u,b,v\}_K, \{v,(q,(a,\$)),z\}_K, v') & \rightarrow T_{q,a\$}(init,v',u,b,z) \\ T_{q,a\$}(x,v',u,b,z), L(x,y), x \neq v', y \neq \bot & \rightarrow L(x,y), T_{q,a\$}(y,v',u,b,z) \\ T_{q,a\$}(x,v',u,b,z), L(x,\bot), x \neq v', \bot \neq v' & \rightarrow \\ \multicolumn{2}{l}{\qquad L(x,v'), L(v',\bot), N_{S2}(\{u,(q',b),v'\}_K, \{v',(a',\$),z\}_K), B_1^t()} \end{array}$$

- for each transition $t = (q,a,q',a',1)$ three role theories are included, each of which consists of three rules:

$$\begin{array}{ll} A_0^t(), N_{R1}(\{u,(q,a),v\}_K, \{v,b,z\}_K, v') & \rightarrow T_{q,a}(init,v',u,b,z) \\ T_{q,a}(x,v',u,b,z), L(x,y), x \neq v', y \neq \bot & \rightarrow L(x,y), T_{q,a}(y,v',u,b,z) \\ T_{q,a}(x,v',u,b,z), L(x,\bot), x \neq v', \bot \neq v' & \rightarrow \\ \multicolumn{2}{l}{\qquad L(x,v'), L(v',\bot), N_{S2}(\{u,a',v'\}_K, \{v',(q',b),z\}_K), A_1^t()} \end{array}$$

and

$$\begin{array}{ll} B_0^t(), N_{R1}(\{u,(q,a),v\}_K, \{v,(b,\$),z\}_K, v') & \rightarrow T_{q,a\$}(init,v',u,b,z) \\ T_{q,a\$}(x,v',u,b,z), L(x,y), x \neq v', y \neq \bot & \rightarrow L(x,y), T_{q,a\$}(y,v',u,b,z) \\ T_{q,a\$}(x,v',u,b,z), L(x,\bot), x \neq v', \bot \neq v' & \rightarrow \\ \multicolumn{2}{l}{\qquad L(x,v'), L(v',\bot), N_{S2}(\{u,a',v'\}_K, \{v',(q',(b,\$)),z\}_K), B_1^t()} \end{array}$$

and

$$\begin{array}{ll} C_0^t(), N_{R2}(\{u,(q,(a,\$)),v\}_K, v') & \rightarrow T_{q,a\$}(init,v',u,\Box,z) \\ T_{q,a\$}(x,v',u,\Box,z), L(x,y), x \neq v', y \neq \bot & \rightarrow L(x,y), T_{q,a\$}(y,v',u,\Box,z) \\ T_{q,a\$}(x,v',u,\Box,z), L(x,\bot), x \neq v', \bot \neq v' & \rightarrow \\ \multicolumn{2}{l}{\qquad L(x,v'), L(v',\bot), N_{S3}(\{u,a',v'\}_K, \{v',(q',(\Box,\$)),z\}_K), C_1^t()} \end{array}$$

- two role theories, each of which consists of exactly one rule:

$$A_0^q(), N_{R3}(\{u,(q,b),v\}_K) \rightarrow N_{S4}(x_0), A_1^q()$$

and

$$B_0^q(), N_{R3}(\{u,(q,(b,\$)),v\}_K) \rightarrow N_{S4}(x_0), B_1^q()$$

are also included for any final state q.

It is clear that any computation of TM on w can be simulated by a run of $\mathcal{P}_{TM,w}$. Moreover, if TM halts on w, then the intruder learns x_0 and $\mathcal{P}_{TM,w}$ is leaky. Conversely, if a run of the protocol reveals the secret x_0, then TM halts on w. □

Remark 2.2 The proof of Theorem 2.3 does not work for protocols in the well-founded form as one can easily see. However, we find the proof interesting by the fact that it shows how, in the absence of freshness check, fresh generation of nonces (or keys) can be simulated with the help of disequality tests. The main ideea is to store in some list all the elements generated up to some step and when a fresh nonce (key) is required it is generated and checked against all the elements in the list.

Theorem 2.3 also shows a new facet of the undecidability of the secrecy problem. All undecidability proofs for the secrecy problem known from the literature on security protocols are based either on finitely many nonces generated by honest agents and arbitrary length messages or infinitely many nonces generated by honest agents and bounded length messages. Our proof shows undecidability of the secrecy problem for protocols with bounded length messages, finitely many nonces generated by honest agents, and infinitely many nonces generated by the intruder.

3 Conclusions

The table below summarizes the main complexity results regarding secrecy for bounded protocols. As we can see, the secrecy problem for bounded protocols without freshness check remains open.

	Bounded protocols	
	without freshness check	with freshness check
initial secrecy	DEXPTIME-complete ([3], Corollary 2.1)	NEXPTIME-complete
secrecy	?	NEXPTIME-complete

The results in Section 2.3 correct Table 9 [4] as follows:

		Unbounded # roles, Bounded ∃ (restricted form)
I with ∃	$\neq$	?
	$=$	NEXPTIME-complete
I no ∃	$\neq$	NEXPTIME-complete
	$=$	NEXPTIME-complete

Acknowledgment The authors are indebted to an anonymous referee who pointed out several subtleties of the multiset rewriting formalism which have led to an improved version of the original submission.

References

[1] W. Charatonik, A. Podelski, J.M. Talbot, Paths vs. Trees in Set-based Program Analysis, *Proceedings of the 27th Annual ACM Symposium on Principles of Programming Languages* 2000, 330–338.

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

[3] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, Undecidability of Bounded Security Protocols, *Workshop on Formal Methods and security Protocols FMSP'99*, Trento (Italy), July 5, 1999.

[4] N. Durgin, P. Lincoln, J. Mitchell, A. Scedrov, Multiset Rewriting and the Complexity of Bounded Security Protocols, *Journal of Computer Security* 12 (2004), 247–311.

[5] R. Ramanujam, S.P. Suresh, A Decidable Subclass of Unbounded Security Protocols, *Proc. of WITS 2003*, April 2003, 11–20.