SP /Fall 2011 /Lecture Notes #03

Bounded Security Protocols

Prof.Dr. Ferucio Laurenţiu Ţiplea

Department of Computer Science

"Al.I.Cuza" University of Iasi

Iasi, Romania

E-mail: fltiplea@mail.dntis.ro

URL: http://www.infoiasi.ro/~fltiplea

Contents

1. Bounded security protocols

Let $\mathcal{P}$ be a protocol, $T \subseteq \mathcal{T}_0$ be a finite set, and $k \geq 1$. A (T, k)-run of $\mathcal{P}$ is any run of $\mathcal{P}$ that satisfies:

- all terms in the run are built up upon T;
- message-length $\leq k$.

$(\mathcal{P}, T, k)$ = protocol under (T, k)-runs or (T, k)-bounded protocol.

The secrecy problem for such protocols is formulated with respect to (T, k)-runs only.

2. Bounds on Bounded Security Protocols

Facts about (T, k)-bounded protocols:

- number of messages bounded by

$$k^2|T|^k = 2^{2\log k + k\log|T|}$$

- number of instantiations bounded by

$$(2^{2\log k + k\log|T|})^{|T|} = 2^{2|T|\log k + k|T|\log|T|}$$

- number of events (action instantiations) bounded by

$$|w| \cdot 2^{2|T|\log k + k|T|\log|T|} = 2^{2|T|\log k + k|T|\log|T| + \log|w|}$$

where w is the sequence of actions in the protocol.

2. Bounds on Bounded Security Protocols

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;
- it takes into consideration the maximum number of bits necessary to represent messages of length at most k.

Remark: the size of the representation of $\mathcal{P}$ is polynomial in $size(\mathcal{P})$, but the number of events that can occur in all (T, k)-runs is exponential in $poly(size(\mathcal{P}))$, for some polynomial $poly$.

3. Case 1: No Freshness Check

Secrecy for bounded protocols without freshness check is in DEXPTIME

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`
  `let` E' be the set of all (T, k)-events; $\xi := \lambda$; $s := s_0$;
  `repeat`
    $E := E'$; $E' := \emptyset$; $bool := 0$;
    `while` $E \neq \emptyset$ `do`
      `begin`
        `choose` $e \in E$; $E := E - \{e\}$;
        `if` $(s, \xi)[e\rangle(s', \xi e)$ `then begin` $s := s'$; $\xi := \xi e$; $bool := 1$ `end`
          `else` $E' := E' \cup \{e\}$;
      `end`
  `until` $bool = 0$;
  `if` $(\bigcup_{A \in Ho} Secret_A) \cap analz(s_I) \neq \emptyset$ `then` "leaky protocol" `else` "non-leaky protocol"
`end.`

3. Case 1: No Freshness Check

The algorithm **A1** terminates in exponential time with respect to the size of the protocol:

- if no event in E can extend to the right the current run ξ, then the algorithm terminates;
- otherwise, all events in E that can extend to the right the current run, taken in an arbitrary but fixed order, are applied in one cycle of `repeat`;
- the number of cycles of `repeat` is bounded by $|E|$;
- at the first cycle of `repeat` the number of cycles in `while` is $|E|$, at the second cycle of `repeat` the number of cycles in `while` is at most $|E|-1$, and so on. Therefore, the number of cycles in `while`, in all `repeat` cycles, is bounded by $|E|+(|E|-1)+\cdots+1=\mathcal{O}(|E|^2)$.

Therefore, the complexity of the algorithm is exponential in $poly(size(\mathcal{P}))$.

3. Case 1: No Freshness Check

Secrecy for bounded protocols without freshness check is DEXPTIME-hard

- reduce the membership problem of unary logic programs, that is DEXPTIME-hard, to the secrecy problem for bounded protocols without freshness check.

3. Case 1: No Freshness Check

Unary logic programs:

- 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, $t_0 \in \{\perp, x, f(x) | f \in \Sigma - \{\perp\}\}$. Moreover, all clauses with $p_0(\perp)$ in the head have only $true$ in the body.

3. Case 1: No Freshness Check

Basic concepts on unary logic programs:

- 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 $true$.

3. Case 1: No Freshness Check

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

> 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.

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).

3. Case 1: No Freshness Check

Theorem 1 The initial secrecy problem for bounded protocols 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;

3. Case 1: No Freshness Check

- 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

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

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

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

3. Case 1: No Freshness Check

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

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

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

3. Case 2: Freshness Check

freshness check – no freshness check

- freshness check:
  - $t \in \overline{s_A \cup M}$ and $M \cap Sub(s) = \emptyset$ (enabling condition)
- no freshness check:
  - $t \in \overline{s_A \cup M}$ (enabling condition)

3. Case 2: Freshness Check

Why **Algorithm A1** does not work in this case?

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

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

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 the algorithm generates a non-leaky maximal run and concludes that the protocol is not leaky.

3. Case 2: Freshness Check

Algorithm A2

`input:` $\mathcal{P}$ a (T, k)-bounded protocol 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.`

For ISP drop the "`for`" statement.

3. Case 2: 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).

Recall that:

$L \in NEXPTIME$ iff $\exists$ a non-deterministic Turing machine TM s.t.

$$L = \{w | TM \ accepts \ w \ in \ time \ 2^{|w|}\}.$$

3. Case 2: Freshness Check

A non-deterministic Turing machine is a tuple

$$TM = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$$

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.

3. Case 2: Freshness Check

Non-deterministic Turing machines working in time 2^n are simulated by bounded security protocols with freshness check as follows:

- a configuration $(q, a_1 \cdots a_n, a, b_1 \cdots b_m)$ is represented by

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

$$\{u_{n+2}, b_1, u_{n+3}\}_K, \ldots, \{u_{n+m+1}, (b_m, \$), u_{n+m+2}\}_K,$$

- initialization on input $w = a_1 \cdots a_n$:

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

3. Case 2: Freshness Check

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

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

or

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

for any $b \in \Gamma$;

3. Case 2: Freshness Check

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

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

or

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

or

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

for any $b \in \Gamma$;

3. Case 2: Freshness Check

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

$$\begin{aligned} A_{q,a}?B_{q,a} &: \{u,(q,a),v\}_K \\ A_{q,a}!B_{q,a} &: x \end{aligned}$$

or

$$\begin{aligned} A_{q,a\$}?B_{q,a\$} &: \{u,(q,(a,\$)),v\}_K \\ A_{q,a\$}!B_{q,a\$} &: x \end{aligned}$$

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

Theorem 2 The (initial) secrecy problem for bounded protocols with freshness check is NEXPTIME-complete.

4. 1-session Bounded Protocols

- A 1-session (T, k)-run of $\mathcal{P}$ is any (T, k)-run of a protocol $\mathcal{P}$ is obtained by applying each role at most once (not necessarily in its entirety), under the same substitution (i.e., all its events are defined by using the same substitution);

- When for the protocol $\mathcal{P}$ only 1-session (T, k)-runs are considered we will say that it is a 1-session (T, k)-bounded protocol;

- A 1-session bounded protocol is a 1-session (T, k)-bounded protocol, for some T and k.

4. 1-session Bounded Protocols

Even if each run of a 1-session bounded protocol is based on exactly one substitution, this substitution is crucial in order to draw the correct conclusion about the secrecy problem. We illustrate this by considering the following 1-session bounded protocol without freshness check:

$$\begin{aligned} A!B &: (\{x, K\})\{x\}_K \\ B?A &: \{x\}_K \\ B!A &: (\{K'\})K' \\ A?B &: K' \end{aligned}$$

In this protocol, A and B are constants, x is a nonce, and K and K' are distinct short-term keys. The 1-session run defined by the identity substitution is non-leaky. However, under the substitution that assigns K to K' and is the identity for all the other elements, the protocol is leaky.

4. 1-session Bounded Protocols

Lemma 1 Let $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ be a 1-session (T, k)-bounded protocol, s_A be the state of an agent A in some run of $\mathcal{P}$, and m be a message of length at most k over T. Then, it is decidable in deterministic polynomial time with respect to $size(\mathcal{P})$ whether $s_A \vDash m$.

Proof Let $D = analz(s_A)$. Decompose m into pieces using analysis rules until either all pieces are in D, or at least one of them is not in D but it cannot be decomposed any further. In the first case $s_A \vDash m$, and in the second case $s_A \nvDash m$. The complexity of this procedure is polynomial in $size(\mathcal{P})$: D can be computed in $\mathcal{O}(size(\mathcal{P})^2)$ (s_A has at most $|w|$ messages of length at most k), m can be decomposed in $\mathcal{O}(size(\mathcal{P}))$, and checking whether each piece in the decomposition of m is in D can be done in $\mathcal{O}(size(\mathcal{P})^3)$ ($|D| = \mathcal{O}(size(\mathcal{P})^2)$). As a conclusion, the complexity of this procedure is $\mathcal{O}(size(\mathcal{P})^3)$. $\square$

4. 1-session Bounded Protocols

The following algorithm shows that (I)SP for 1-session bounded protocols with or without freshness check is in NP.

Algorithm A3

`input:` $\mathcal{P} = (\mathcal{S}, \mathcal{C}, w)$ a 1-session (T, k)-bounded protocol;
`output:` "leaky protocol" if $\mathcal{P}$ has leaky 1-session (T, k)-runs;
`begin`
  `guess` a 1-session sequence $\xi = e_1 \cdots e_n$ of $\mathcal{P}$ under some substitution σ;
  `if` ξ is a 1-session run `then`
    `begin`
      `let` $s_0[e_1\rangle s_1[\cdots[e_n\rangle s_n$; $Secret := \bigcup_{A \in Ho} Secret_A$;
      `for` $i := 1$ `to` $n - 1$ `do`
          $Secret := Secret \cup ((\bigcup_{A \in Ho} analz(s_{iA})) - analz(s_{iI}))$;
      `if` $Secret \cap analz(s_{nI}) \neq \emptyset$ `then` "leaky protocol"
    `end`
`end.`

For ISP drop the "`for`" statement.

4. 1-session Bounded Protocols

The algorithm **A3** terminates in non-deterministic polynomial time with respect to $poly(size(\mathcal{P}))$ and, therefore, SP for 1-session bounded protocols with or without freshness check is in NP.

To prove that SP for 1-session bounded protocols with or without freshness check is NP-hard, a reduction from 3-SAT is exhibited.

An instance of 3-SAT consists of a set of boolean variables $\{x_1, \ldots, x_n\}$ and a conjunction $E = C_1 \wedge \cdots \wedge C_m$ of *clauses* , where each clause $C_j = \alpha_{j,1} \vee \alpha_{j,2} \vee \alpha_{j,3}$ is a disjunction of *literals* $\alpha_{j,l}$, and each literal $\alpha_{j,l}$ is either x or $\neg x$, for some variable x. E is *satisfiable* if it can be evaluated to the truth value true under some assignment.

4. 1-session Bounded Protocols

Let E be a 3-SAT instance as the one above. A 1-session (T, k)-bounded protocol $\mathcal{P}_{3SAT}$ will be defined such that E is satisfiable if and only if the protocol $\mathcal{P}_{3SAT}$ reveals secrets generated in the course of some run.

The idea of the construction is to let the intruder generate an assignment as a possible solution to the 3-SAT instance E. After generating such an assignment, the intruder will use the honest participants to check the truth value of each clause of E. If the assignment generated by the intruder is a solution, then a secret will be generated and revealed by some honest agent.

4. 1-session Bounded Protocols

The protocol:

- A, B, A_i, B_i, $A_{j,l}$, and $B_{j,l}$ are pairwise distinct agents, for any $1 \leq i \leq n$, $1 \leq j \leq m$, and $l = 1, 2, 3$. A_i and B_i correspond to the variable x_i, and $A_{j,l}$ and $B_{j,l}$ correspond to the literal $\alpha_{j,l}$;
- V_i is a long-term key shared by A_i, B_i, $A_{j,l}$, and $B_{j,l}$ such that x_i defines the literal $\alpha_{j,l}$ (that is, $\alpha_{j,l}$ is either x_i or $\neg x_i$), for any i, j, and l;
- $K_{j,A}$ is a long-term key shared by $A_{j,l}$ and A, for any j and l;
- K is a long-term key shared by all honest agents;
- $True$ and $False$ are two distinct constant nonces representing the truth values *true* and, respectively, *false*. They will be used by the intruder to generate an assignment. We also consider v and x two distinct nonces;

4. 1-session Bounded Protocols

- the protocol actions are:
  - (variable assignment)

$$\begin{aligned} A_i?B_i &: v \\ A_i!B_i &: \{v\}_{V_i} \end{aligned}$$

for each $1 \leq i \leq n$.

When A_i receives (from the intruder) a truth value v, he assigns it to x_i by sending $\{v\}_{V_i}$. As there is exactly one value for each i, any 1-session run will ensure that all variables are instantiated in a non-redundant way;

4. 1-session Bounded Protocols

– (checking the truth value of clause C_j)

$$\begin{aligned} A_{j,l}?B_{j,l} &: \{True\}_{V_i} \\ A_{j,l}!B_{j,l} &: \{True\}_{K_{j,A}} \end{aligned}$$

if $\alpha_{j,l} = x_i$, and

$$\begin{aligned} A_{j,l}?B_{j,l} &: \{False\}_{V_i} \\ A_{j,l}!B_{j,l} &: \{True\}_{K_{j,A}} \end{aligned}$$

if $\alpha_{j,l} = \neg x_i$, for each j and l.

A clause C_j is satisfiable iff at least one of its literals is evaluated to the truth value $true$. Therefore, if an agent $A_{j,l}$ receives an assignment $\{True\}_{V_i}$ for a variable x_i and $\alpha_{j,l} = x_i$, then it returns $\{True\}_{K_{j,A}}$ because, from his point of view, C_j is evaluated to the truth value *true*.

4. 1-session Bounded Protocols

– (reveal the secret)

$$\begin{aligned} A?B &: \{True\}_{K_{1,A}}, \ldots, \{True\}_{K_{m,A}} \\ A!B &: (\{x\})\{x\}_K \\ A!B &: x \end{aligned}$$

When A receives the confirmation that all clauses were evaluated to the truth value true, then A generates a secret x, sends x encrypted by K, and then reveals x.

4. 1-session Bounded Protocols

Let T be the set of all basic terms of the protocol $\mathcal{P}_{3SAT}$. Then, it is easy to see that E is satisfiable if and only if there exists a 1-session (T, k)-run of $\mathcal{P}_{3SAT}$ which reveals a secret generated in the course of the run, where k is some suitable chosen constant linear in m. Moreover, the size of $\mathcal{P}_{3SAT}$ is polynomial in $max\{n, m\}$. Therefore, 3-SAT can be reduced in polynomial time to SP.
In order to reduce 3-SAT to ISP what we have to do is to provide A in the protocol above with some initial secret x_0 and to replace the last two actions by "$A!B : x_0$".

Theorem 3 The (initial) secrecy problem for 1-session bounded protocols with or without freshness check is NP-complete.

5. Relationship with the MSR Formalism

The MSR (multiset rewriting) uses existentials and disequality tests:

- existential = freshness check;
- disequality tests add more power to the MSR formalism and they do not have any equivalent in the formalism in this talk;
- intruder with no existentials = intruder does not generate new elements;
- intruder with existentials = intruder can generate new elements (it does not have any equivalent in the formalism used in this paper);
- intruder with bounded existentials = intruder can generate new elements from a finite set of elements (it does not have any equivalent in the formalism used in this paper)
- protocol with bounded existentials = honest agents can generate new elements from a given finite set.

5. Relationship with the MSR Formalism

As a conclusion of our discussion above we can say that any protocol in the formalism in this talk 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 any existentials and disequality tests;
- if the protocol is with freshness check then the corresponding protocol in the MSR formalism may have existentials but it does not have any disequality tests.

5. Relationship with the MSR Formalism

Corollary 2 The following four problems, in the MSR formalism, are NEXPTIME-complete:

1. ISP for protocols with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials;
2. ISP for protocols with bounded length messages, bounded existentials, with disequality tests, and an intruder with no existentials;
3. ISP for protocols with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials.
4. ISP for protocols with bounded length messages, bounded existentials, with disequality tests, and an intruder with bounded existentials.

5. Relationship with the MSR Formalism

Corollary 2 corrects three false statements in

- 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.

according to which:

- ISP for protocols with bounded length messages, bounded existentials, with disequality tests, and an intruder with no existentials,
- ISP for protocols with bounded length messages, bounded existentials, without disequality tests, and an intruder with no existentials,
- ISP for protocols with bounded length messages, bounded existentials, without disequality tests, and an intruder with existentials

are DEXPTIME-complete.

5. Relationship with the MSR Formalism

The source of these mistakes was that the above authors claimed that the Algorithm A1 works in the case of existentials, which is false.

By the technique in the proof of Theorem 2 we also get:

Theorem 4 ISP for protocols with bounded length messages, no existentials, disequality tests and an intruder with existentials is undecidable.

Corollary 3 ISP for protocols with bounded length messages, bounded existentials, disequality tests and an intruder with existentials is undecidable.

6. Summary of Results

Bounded Protocols	1-session		Multi-session	
	ISP	SP	ISP	SP
Freshness check	NP-c [RuTu2003]	NP-c	NEXP-c	NEXP-c
No freshness check	NP-c [DLMS2004]	NP-c	DEXP-c [DLMS1999]	?

Column four in Table 9 in [DLMS2004] should be:

Bounded Protocols		Unbounded # roles
		bounded $\exists$
I with $\exists$	$\neq$	undecidable
	$=$	NEXP-c
I no $\exists$	$\neq$	NEXP-c
	$=$	NEXP-c