# Detecting Replay Attacks by Freshness Annotations *

Han Gao[1], Pierpaolo Degano[2], Chiara Bodei[2], and Hanne Riis Nielson[1]

[1] Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads bldg 322, DK-2800 Kongens Lyngby, Denmark. {hg,riis}@imm.dtu.dk

[2] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo, 3, I-56127, Pisa, Italy. {degano,chiara}@di.unipi.it

**Abstract.** We present a reduction semantics for the LySA calculus extended with session information and a static analysis for it. If a protocol passes the analysis then it is free of replay attacks.

## 1 Introduction

Since the 80's, formal analyses of cryptographic protocols have been widely studied. Many formal methods such as BAN logic [4], model checking and theorem proving have been put forward. The formal model for security protocols built by Dolev and Yao is particularly significant. Indeed, most of the formal analysis tools were built upon it, e.g. Meadows and Syverson NRL [7], Millen Interrogator [8], Paulson inductive method [12], Isabelle [13], etc. Each tool is equipped to detect a certain amount of attacks, including replay attacks.

Replay attacks are classified, by Syverson in [14], at the highest level as run-external and run-internal attacks, depending on the origin of messages.

In this paper, we restrict our attention to run-external attacks. This type of attacks allow the attacker to achieve messages from one run of a protocol, often referred to as a *session*, and to send them to a principal participating in another run of the protocol.

Here we extend the LySA calculus [2, 3] with annotations about sessions. A control flow analysis is proposed for the extended LySA, which soundly over-approximates the behavior of protocols. It tracks the set of messages that are transferred over the network, and records the potential values of variables. Since our analysis is sound, we capture malicious activities, expressed in terms of annotation violations. In contrast to model checking approaches, our static analysis is fully automatic and termination is always guaranteed. The proposed analysis has been implemented. The resulting tool was applied to some cryptographic protocols, such as Otway-Rees [11] and Needham-Schroeder [10].

The paper is organized as follows. In Section 2, we present the LySA calculus annotated with session information. We introduce the control flow analysis in Section 3 and prove our main results. In Section 4 we conclude with an assessment of our approach.

* This work has been partially supported by the project SENSORIA.

## 2 A reduction semantics for the LySA calculus

LySA [2, 3] is a process algebra, in the tradition of the $\pi$- [9] and Spi- [1] calculi. Among its peculiar features, there are: (1) the absence of channels: in LySA all processes have only access to a single global communication channel, the ether and (2) tests associated with input and decryption are expressed using pattern matching.

### 2.1 Syntax

LySA consists of terms and processes. The syntax of terms $E$ and processes $P$ is given below. Here $\mathcal{N}$ and $\mathcal{X}$ denote sets of names and variables, respectively. For the sake of simplicity, we only consider here some basic terms and encryptions. The name $n$ is used to represent keys, challenges and names of principals. Encryptions are tuples of terms $E_1, \ldots, E_k$ encrypted under a shared key represented by the term $E_0$. We adopt an assumption of perfect cryptography in this paper.

| | |
|---|---|
| $E ::=$ | *standard terms* |
| $n$ | name ($n \in \mathcal{N}$) |
| $x$ | variable ($x \in \mathcal{X}$) |
| $\{E_1, \ldots, E_k\}_{E_0}$ | symmetric encryption ($k \geq 0$) |
| | |
| $P ::=$ | *standard processes* |
| $0$ | nil |
| $P_1 \mid P_2$ | parallel composition |
| $(\nu\ n)P$ | restriction |
| $!P$ | replication |
| *decrypt* $E$ *as* $\{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}_{E_0}$ *in* $P$ | symmetric decryption |
| $(E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P$ | input |
| $\langle E_1, \ldots, E_k \rangle.P$ | output |

In addition to the classical constructs for composing processes, LySA also contains an input construct with matching and a decryption operation with matching. The idea behind the matching is as follows: we allow a prefix of the received tuple to match a selection of values. If the test is passed, the remaining values are bound to the relevant variables.

**Extended LySA** We change the syntax of standard LySA so that each term and process now carries an identifier of the session it belongs to. In what follows, we assume that $SID$ is a fixed enumerable set of session identifiers $s$, and denote $\mathcal{E}_1, \mathcal{E}_2, \ldots$ the extended terms and $\mathcal{P}, \mathcal{Q}, \ldots$ the extended processes defined below. Note that variables carry no annotation and therefore we shall consider $[x]_s$ and $x$ to be the same (see below).

| | |
|---|---|
| $\mathcal{E} ::=$ | *terms with session identifiers* |
| $[n]_s$ | name |
| $x$ | variable |
| $[\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s$ | symmetric encryption ($k \geq 0$) |
| | |
| $\mathcal{P} ::=$ | *processes with session identifiers* |
| $0$ | nil |
| $\mathcal{P}_1 \mid \mathcal{P}_2$ | parallel composition |
| $(\nu\ [n]_s)\mathcal{P}$ | restriction |
| $[!P]_s$ | replication |
| *decrypt* $\mathcal{E}$ *as* $\{\mathcal{E}_1, \ldots, \mathcal{E}_j; x_{j+1}, \ldots, x_k\}_{\mathcal{E}_0}$ *in* $\mathcal{P}$ | symmetric decryption |
| $(\mathcal{E}_1, \ldots, \mathcal{E}_j; x_{j+1}, \ldots, x_k).\mathcal{P}$ | input |
| $\langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle.\mathcal{P}$ | output |

We define a function $\mathcal{F}$ and a function $\mathcal{T}$, in the style of [5], that map standard terms and processes into the extended ones, by attaching the session identifiers inductively. Note that $\mathcal{F}$ unwinds the syntactic structure of an extended term until reaching a basic term (a name or a variable), while $\mathcal{T}$ unwinds the structure of an extended process until reaching a nil (which is untagged) or a replication.

**Definition 1.** *Distributing Session Identifiers*

$\mathcal{F} : E \times s \to \mathcal{E}$

$-\mathcal{F}([n]_s) = [n]_s \qquad -\mathcal{F}([x]_s) = x$

$-\mathcal{F}([\{E_1, \ldots, E_k\}_{E_0}]_s) = [\{\mathcal{F}([E_1]_s), \ldots, \mathcal{F}([E_k]_s)\}_{\mathcal{F}([E_0]_s)}]_s$

$\mathcal{T} : P \times s \to \mathcal{P}$

$-\mathcal{T}([\langle E_1, \ldots, E_k \rangle.P]_s) = \langle \mathcal{F}([E_1]_s), \ldots, \mathcal{F}([E_k]_s) \rangle.\mathcal{T}([P]_s)$

$-\mathcal{T}([(E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P]_s) =$
$(\mathcal{F}([E_1]_s), \ldots, \mathcal{F}([E_j]_s); x_{j+1}, \ldots, x_k).\mathcal{T}([P]_s)$

$-\mathcal{T}([\textit{decrypt } E \textit{ as } \{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}_{E_0} \textit{ in } P]_s) =$
$\textit{decrypt } \mathcal{F}([E]_{sid}) \textit{ as } \{\mathcal{F}([E_1]_s), \ldots, \mathcal{F}([E_j]_s); x_{j+1}, \ldots, x_k\}_{\mathcal{F}([E_0]_s)} \textit{ in } \mathcal{T}([P]_s)$

$-\mathcal{T}([P \mid Q]_s) = \mathcal{T}([P]_s) \mid \mathcal{T}([Q]_s) \qquad -\mathcal{T}([(\nu\ n)P]_s) = (\nu\ [n]_s)\mathcal{T}([P]_s)$

$-\mathcal{T}([!P]_s) = [!P]_s \qquad -\mathcal{T}([0]_s) = 0$

For our subsequent treatment, it is convenient introducing two auxiliary operators. The first one, $\approx$, defines the equivalence between extended terms, i.e. two extended terms are equal if and only if their standard counterparts are equal and the second one, $\mathcal{I}$, extracts the outermost session identifier of an extended term. Note that $\mathcal{I}$ is only defined on closed terms.

**Definition 2.** *Two auxiliary operators*

- *Assume* $\mathcal{F}([E_1]_{s1}) = \mathcal{E}_1$ *and* $\mathcal{F}([E_2]_{s2}) = \mathcal{E}_2$*, then* $\mathcal{E}_1 \approx \mathcal{E}_2$ *iff* $E_1 = E_2$
- $\mathcal{I}$*:* $\mathcal{E} \to SID$ *such that: (i)* $\mathcal{I}([n]_s) = s$*; (ii)* $\mathcal{I}([\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s) = s$

### 2.2 Operational Semantics

Below we assume the standard *structural congruence* $\equiv$ on LySA processes, as the least congruence satisfying the following clauses (as usual $fn(P)$ is the set of the free names of $P$):

1. $P \equiv Q$ if $P$ and $Q$ are disciplined $\alpha-$equivalent
2. $P \mid 0 \equiv P$
3. $P \mid Q \equiv Q \mid P$
4. $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
5. $(\nu x)0 \equiv 0$
6. $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
7. $(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q \; if \; x \notin fn(P)$

To simplify the definition of our control flow analysis in Section 3, we discipline the $\alpha$-renaming of bound values and variables. To do it in a simple and "implicit" way, we assume that values and variables are "stable", i.e. that for each value $n \in \mathcal{N}$ there is a canonical representative $\lfloor n \rfloor$ for the set $\{n, n_0, n_1, \cdots\}$ and similarly, for each variable $x \in \mathcal{X}$ there is a canonical representative $\lfloor x \rfloor$ for the set $\{x, x_0, x_1, \cdots\}$. Then, we discipline $\alpha$-conversion as follows: two values (resp. variables) are $\alpha$-convertible only when they have the same canonical value (resp. variable). In this way, we statically maintain the identity of values and variables that may be lost by freely applying $\alpha$-conversions. Hereafter, we shall simply write $n$ (resp. $x$) for $\lfloor n \rfloor$ (resp. $\lfloor x \rfloor$).

Following the tradition of the $\pi$-calculus, we shall give the extended LySA a reduction semantics. The *reduction relation* $\xrightarrow{\alpha}_{\mathcal{R}}$ is the least relation on closed processes that satisfies the rules in Table 1. It uses the standard notion of substitution, $\mathcal{P}[\mathcal{E}/x]$, structural congruence, as defined above, and the disciplined treatment of $\alpha$-conversion. The reduction relation carries labels on the arrows to record some key movements during the computational steps.

As far as the semantics is concerned, we consider two variants of *reduction relation* $\xrightarrow{\alpha}_{\mathcal{R}}$, graphically identified by a different instantiation of the relation $\mathcal{R}$, which decorates the transition relation. One variant ( $\xrightarrow{\alpha}_{\mathsf{RM}}$) takes advantage of annotations, the other one ( $\xrightarrow{\alpha}$) discards them: essentially, the first semantics checks the freshness of messages, while the other one does not (see below):

- the *reference monitor semantics* $\mathcal{P} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{Q}$ takes

$$\mathbf{RM}((s_1, s_1'), \ldots, (s_k, s_k')) = \vee_{i=1}^{k} \;\; (s_i = s_i')$$

- the *standard semantics* $\mathcal{P} \xrightarrow{\alpha} \mathcal{Q}$ takes, by construction, $\mathcal{R}$ to be universally true.

The rule (Com) expresses that an output $\langle \mathcal{E}_1, \ldots, \mathcal{E}_j, \mathcal{E}_{j+1}, \ldots, \mathcal{E}_k \rangle.\mathcal{P}$ is matched by an input $(\mathcal{E}_1, \ldots, \mathcal{E}_j; x_{j+1}, \ldots, x_k)$ in case the first $j$ elements are pairwise the same when all the annotations are recursively removed. When the matchings are successful each $E_i$ is bound to the corresponding $x_i$.

$$(Com)\quad \frac{\wedge_{i=1}^{j}\ \mathcal{E}_i \approx \mathcal{E}'_i}{\langle \mathcal{E}_1,\ldots,\mathcal{E}_k\rangle.\mathcal{P} \mid (\mathcal{E}'_1,\ldots,\mathcal{E}'_j; x_{j+1},\ldots,x_k).\mathcal{Q} \xrightarrow{\langle \mathcal{E}_1,\ldots,\mathcal{E}_k\rangle}_{\mathcal{R}} \mathcal{P} \mid \mathcal{Q}[\mathcal{E}_{j+1}/x_{j+1},\ldots,\mathcal{E}_k/x_k]}$$

$$(Decr)\quad \frac{\wedge_{i=0}^{j}\ \mathcal{E}_i \approx \mathcal{E}'_i\ \wedge\ \mathcal{R}((\mathcal{I}(\mathcal{E}_0),\mathcal{I}(\mathcal{E}'_0)),\ldots,(\mathcal{I}(\mathcal{E}_j),\mathcal{I}(\mathcal{E}'_j)))}{\mathit{decrypt}\ [\{\mathcal{E}_1,\ldots,\mathcal{E}_k\}_{\mathcal{E}_0}]_s\ \mathit{as}\ \{\mathcal{E}'_1,\ldots,\mathcal{E}'_j; x_{j+1},\ldots,x_k\}_{\mathcal{E}'_0}\ \mathit{in}\ \mathcal{P} \xrightarrow{\{\mathcal{E}_1,\ldots,\mathcal{E}_k\}_{\mathcal{E}_0}}_{\mathcal{R}} \mathcal{P}[\mathcal{E}_{j+1}/x_{j+1},\ldots,\mathcal{E}_k/x_k]}$$

$$(Par)\quad \frac{\mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{P}'}{\mathcal{P} \mid \mathcal{Q} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{P}' \mid \mathcal{Q}} \qquad (Congr)\quad \frac{P \equiv Q\ \wedge\ \mathcal{T}([Q]_s) \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{Q}'}{\mathcal{T}([P]_s) \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{Q}'}$$

$$(Res)\quad \frac{\mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{P}'}{(\nu\ [n]_s)\mathcal{P} \xrightarrow{(\nu\ [n]_s)\alpha}_{\mathcal{R}} (\nu\ [n]_s)\mathcal{P}'} \qquad (Repl)\quad [!P]_s \xrightarrow{\iota}_{\mathcal{R}} \mathcal{T}([P]_s) \mid [!P]_{s'}\ \ (s' \text{ fresh})$$

**Table 1.** Operational Semantics $\mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{P}'$, parameterized on $\mathcal{R}$

Similarly, the rule (Decr) expresses the result of matching an encryption $[\{\mathcal{E}_1,\ldots,\mathcal{E}_k\}_{\mathcal{E}_0}]_s$ with $\mathsf{decrypt}\ \mathcal{E}\ \mathsf{as}\ \{\mathcal{E}'_1,\ldots,\mathcal{E}'_j; x_{j+1},\ldots,x_k\}_{\mathcal{E}'_0}\ \mathsf{in}\ \mathcal{P}$. As was the case for communication, the first $j$ components $\mathcal{E}_i$ and $\mathcal{E}_i$ must be "equal" in terms of $\approx$, and additionally the keys must be the same, i.e. $\mathcal{E}_0 \approx \mathcal{E}'_0$. When the matching is successful, each $\mathcal{E}_i$ is bound to the corresponding $x_i$. In the *reference monitor semantics* we ensure that the decrypted message comes from the current session by checking whether the first $j$ components $\mathcal{E}_i$ and $\mathcal{E}'_i$ have the same session identifiers. In the *standard semantics* the condition $\mathcal{R}((\mathcal{I}(\mathcal{E}_0),\mathcal{I}(\mathcal{E}'_0)),\ldots,(\mathcal{I}(\mathcal{E}_j),\mathcal{I}(\mathcal{E}'_j)))$ is universally true and thus can be ignored.

The rule (Congr) makes use of the function $\mathcal{T}$, which bridges the gap between the semantics defined on the extended processes $\mathcal{P}$ and the structural congruence defined on the standard processes $P$.

In case of (Res), the restricted names are re-arranged in such a way that they are put in outermost position, by enlarging their scope and possibly $\alpha-$converting names. The re-arrangement is defined as

$$(\nu[n]_s)\alpha = \begin{cases} (\nu(\vec{m},[n]_s))\alpha' & \text{if } \alpha = (\nu\vec{m})\alpha' \\ (\nu([n]_s,\vec{m}))\alpha' & \text{if } \alpha = \alpha'(\nu\vec{m}) \end{cases}$$

where $\vec{m}$ stands for a list of names with session identifiers, e.g. $[m_1]_{s_1}, [m_2]_{s_2}, \ldots$.

In case of (Repl), the process makes a silent move $\iota$ and is unfolded once. Note that the new session identifier, $s'$ in this case, has to be unique for not mixing the processes up.

The rule (Par) is standard.

### 2.3 Dynamic Property

As for the dynamic property of the process, we shall consider the names localized to their generators and define the property *fresh* as:

**Definition 3.** *A process $\mathcal{P}$ enjoys the* freshness *property iff for all the derivations of $\mathcal{P}$*

$$\mathcal{P} \xrightarrow{\alpha_1}_{\mathcal{R}} \ldots \xrightarrow{\alpha_l}_{\mathcal{R}} \mathcal{T}([R]_s) \mid \mathcal{Q} \xrightarrow{(\nu\vec{m})\{\mathcal{E}_1,\ldots,\mathcal{E}_k\}\varepsilon_0}_{\mathcal{R}} \mathcal{T}([R']_s) \mid \mathcal{Q}$$

*there is at least one term $\mathcal{E}_i$ $(1 \leq i \leq k)$ in the encryption $\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}$ such that $\mathcal{E}_i \in \vec{m}$ and $\mathcal{I}(\mathcal{E}_i) = s$.*

### 2.4 Example

We shall use the Wide Mouthed Frog protocol [4] (WMF) to illustrate how to encode protocols in our calculus. WMF is a symmetric key management protocol aiming at establishing a secret session key $K_{ab}$ between the two principals $A$ and $B$ sharing secret master keys $K_A$ and $K_B$, respectively, with a trusted server $S$. The protocol is specified by the following informal narration:

$$\begin{array}{l} 1.\ A \rightarrow S : \{B, K_{ab}\}_{K_A} \\ 2.\ S \rightarrow B : \{A, K_{ab}\}_{K_B} \\ 3.\ B \rightarrow A : \{Msg\}_{K_{ab}} \end{array}$$

In the first message $A$ sends to $S$ its name, and then a fresh key $K_{ab}$ and the name of the intended receiver $B$, encrypted under the key $K_A$. In the second one, $S$ forwards the key and the sender name $A$ to $B$, encrypted under the key $K_B$. Finally, $B$ sends $A$ the message $Msg$ encrypted under the session key $K_{ab}$ (Note that usually $A$ sends the last message).

The extended LySA specification of the WMF protocol is $[!P]_0$ where $P = (A|B|S)$ contains three processes running in parallel, each of them models one principal's activity, and is as follows:

$$\begin{array}{lll|l} 1. & A & & (\upsilon\ K_{ab}) \\ & A \rightarrow & & \langle A, S, \{B, K_{ab}\}_{K_A}\rangle. \\ 3'. & \rightarrow A & & (B, A; z). \\ 3''. & & A & \mathit{decrypt}\ z\ \mathit{as}\ \{; z_m\}_k\ \mathit{in}\ 0 \\ 2'. & \rightarrow B & \mid & (S, B; y). \\ 2''. & & B & \mathit{decrypt}\ y\ \mathit{as}\ \{A; k\}_{K_B}\ \mathit{in} \\ 3. & B & & (\upsilon\ Msg) \\ & B \rightarrow & & \langle B, A, \{Msg\}_k\rangle.0 \\ 1'. & \rightarrow S & \mid & (A, S; p). \\ 1''. & & S & \mathit{decrypt}\ p\ \mathit{as}\ \{B; k'\}_{K_A}\ \mathit{in} \\ 2. & S \rightarrow & & \langle S, B, \{A, k'\}_{K_B}\rangle.0 \end{array}$$

## 3 Static Analysis

The aim of the analysis is to give a safe over-approximation of all possible behavior of protocols; this will include the possible messages communicated over the network and the possible value-bindings of the variables. At each decryption, the analysis will check whether the freshness property is preserved and any possible violation will be regarded in an error component of the analysis.

### 3.1 Analysis of terms

For each term $\mathcal{E}$, the analysis will determine a superset of the possible values it may evaluate to. For this we keep track of the potential values of variables and to this end we introduce a global *abstract environment*:

- $\rho : \mathcal{X} \to \wp(\mathcal{V})$ maps the variables to the sets of values that they may be bound to.

where $\mathcal{V}$ is for the set of terms with no free variables. The judgement for expressions takes the form $\rho \models \mathcal{E} : \vartheta$ where $\vartheta \subseteq \mathcal{V}$ is an acceptable estimate of the set of values that $\mathcal{E}$ may evaluate to in the environment $\rho$. The judgement is defined by the axioms and rules of Table 2. Basically, the rules amount to demanding the $\vartheta$ contains all the values associated with the components of a term. In the sequel we shall use two kinds of membership tests: the usual $V \in \vartheta$ that simply tests whether $V$ is in the set $\vartheta$ and the *faithful* test $V \propto \vartheta$ that holds if there is a value $V'$ in $\vartheta$ that equals $V$ when the annotations are inductively ignored.

### 3.2 Analysis of processes

In the analysis of processes we focus on which values may flow on the network:

- $\kappa \subseteq \wp(\mathcal{V}^*)$: the *abstract network environment* that includes all the message sequences that may flow on the network.

The judgement for processes has the form: $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$ expressing that $\rho, \kappa$ and $\psi$ are valid analysis estimates of process $\mathcal{P}$, where $\psi$ is the possibly empty set of error-component which collects an over-approximation of the freshness violations; we prove in Theorem 2 (in Section 3.1) that when $\psi = \emptyset$ we may dispense with the reference monitor. The judgement is defined by the axioms and rules in the lower part of Table 2 and is explained below.

The rule for *output* does two things: first, all the expressions are evaluated and then it is required that all the combination of the values found by this evaluation is recorded in $\kappa$. Finally, the continuation process must be analysed.

The rule for *input* incorporates pattern matching, which is dealt with by first evaluating all the of first $j$ expressions in the input to be the sets $\vartheta_i$ for $i = 1, \ldots, j$. Next, if any of the sequences of length $k$ in $\kappa$ are such that the first $j$ values component-wise are included in $\vartheta_i$ then the match is concluded to be

$$\frac{[n]_s \in \vartheta}{\rho \models [n]_s : \vartheta} \qquad \frac{\rho(x) \subseteq \vartheta}{\rho \models x : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=0}^{k} \rho \models \mathcal{E}_i : \vartheta_i \ \wedge \\ \forall v_0, \ldots, v_k : \wedge_{i=0}^{k} v_i \in \vartheta_i \Rightarrow [\{v_1, \ldots, v_k\}_{v_0}]_s \in \vartheta\end{array}}{\rho \models [\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s : \vartheta}$$

$$\frac{\begin{array}{c}\wedge_{i=1}^{k} \rho \models \mathcal{E}_i : \vartheta_i \ \wedge \ \forall v_1, \ldots, v_k \wedge_{i=1}^{k} v_i \in \vartheta_i \Rightarrow \\ \langle v_1, \ldots, v_k \rangle \in \kappa \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} \mathcal{P}\end{array}}{\rho, \kappa \models_{\mathsf{RM}} \langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle.\mathcal{P}}$$

$$\frac{\begin{array}{r}\rho \models \mathcal{E} : \vartheta \ \wedge \ \wedge_{i=0}^{j} \rho \models \mathcal{E}_i : \vartheta_i \wedge \forall [\{v_1, \ldots, v_k\}_{v_0}]_s \in \vartheta : \wedge_{i=0}^{j} v_i \propto \vartheta_i \Rightarrow \\ \wedge_{i=j+1}^{k} v_i \in \rho(x_i) \ \wedge \\ (\nexists i : 1 \leq i \leq k : \mathcal{I}(v_i) \neq \mathcal{I}(\mathcal{E}_i) \Rightarrow (x_{j+1}, \ldots, x_k) \in \psi) \ \wedge \\ \rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi\end{array}}{\rho, \kappa \models_{\mathsf{RM}} \text{decrypt } \mathcal{E} \text{ as } \{\mathcal{E}_1, \ldots, \mathcal{E}_j; x_{j+1}, \ldots, x_k\}_{\mathcal{E}_0} \text{ in } \mathcal{P} : \psi}$$

$$\frac{\begin{array}{c}\wedge_{i=1}^{j} \rho \models \mathcal{E}_i : \vartheta_i \ \wedge \ \forall \langle v_1, \ldots, v_k \rangle \in \kappa : \wedge_{i=i}^{j} v_i \propto \vartheta_i \Rightarrow \\ \wedge_{i=j+1}^{k} v_i \in \rho(x_i) \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi\end{array}}{\rho, \kappa \models_{\mathsf{RM}} (\mathcal{E}_1, \ldots, \mathcal{E}_j; x_{j+1}, \ldots, x_k).\mathcal{P} : \psi}$$

$$\frac{\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) : \psi \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_{s'}) : \psi}{\rho, \kappa \models_{\mathsf{RM}} [!P]_s : \psi}$$

$$\frac{\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} \mathcal{Q} : \psi}{\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} \mid \mathcal{Q} : \psi}$$

$$\rho, \kappa \models_{\mathsf{RM}} 0 : \psi \qquad \frac{\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi}{\rho, \kappa \models_{\mathsf{RM}} (\nu [n]_s)\mathcal{P} : \psi}$$

**Table 2.** Analysis of terms and processes

successful. In this case, the remaining values of the $k$-tuple must be recorded in $\rho$ as possible bindings of the variables.

The rule for *decryption* handles the matching similarly to the rule for *input*. The only difference is that here the matching is performed against $j + 1$ component including the key. We use the faithful test for matching because the semantics ignores the annotations. After the successful matching, values are bound to the corresponding variables and, more important, the session identifiers of the key and first $j$ components have to be checked for equivalence. In case of unequal, meaning that the values are not from the current session, the $x_i$ is recorded in the error component $\psi$.

The rule for *replication* attaches two different session identifiers to two copies of the process before analysing both of them. Again the newly generated session identifier has to be unique in order not to mix processes up. We prove in Theorem 1 (in Section 3.1) that it is enough to only analyse two copies of the process.

The rules for the inactive process, parallel composition and restriction are straightforward.

### 3.3 Example

We analyse the WMF protocol and the least solution has a non-empty $\psi$-component, i.e.

$$\rho, \kappa \models_{\mathsf{RM}} WMF : \psi$$

where $\rho$, $\kappa$ and $\psi$ have these non-empty entries

$$\begin{aligned}
\rho : \; & y \mapsto \{\{[A]_0, [K_{ab}]_0\}_{[K_B]_0}, \{[A]_1, [K_{ab}]_1\}_{[K_B]_1}\} \\
& z \mapsto \{\{[Msg]_0\}_{[K_{ab}]_0}, \{[Msg]_1\}_{[K_{ab}]_1}\} \\
& p \mapsto \{\{[B]_0, [K_{ab}]_0\}_{[K_A]_0}, \{[B]_1, [K_{ab}]_1\}_{[K_A]_1}\} \\
& k \mapsto \{[K_{ab}]_0, [K_{ab}]_1\} \\
& k' \mapsto \{[K_{ab}]_0, [K_{ab}]_1\} \\
& z_m \mapsto \{[Msg]_0, [Msg]_1\} \\
\kappa : \; & \{\langle [A]_0, [S]_0, [\{[B]_0, [K_{ab}]_0\}_{[K_A]_0}]_0\rangle, \langle [A]_1, [S]_1, [\{[B]_1, [K_{ab}]_1\}_{[K_A]_1}]_1\rangle\} \cup \\
& \{\langle [B]_0, [A]_0, [\{[Msg]_0\}_{[K_{ab}]_0}]_0\rangle, \langle [B]_1, [A]_1, [\{[Msg]_1\}_{[K_{ab}]_1}]_1\rangle\} \cup \\
& \{\langle [S]_0, [B]_0, [\{[A]_0, [K_{ab}]_0\}_{[K_B]_0}]_0\rangle, \langle [S]_1, [B]_1, [\{[A]_1, [K_{ab}]_1\}_{[K_B]_1}]_1\rangle\} \\
\psi : \; & \{(k), (k'), (z_m)\}
\end{aligned}$$

According the rule for $[!P]_0$ in Table 2, the analysis makes two copies of $P$ with different session identifiers (0 and 1 in our case), which models two sessions running together.

Because the messages from both sessions are sent over the same network, which the attacker has the total control of, the attacher can then fool a principal to accept a message actually from another session. This is suggested by the non-empty $\psi$: the three variables in $\psi$ indicate that messages in step $1''$, $2''$ and $3''$ may not be fresh. This is highly dangerous because the principal may be forced to use an old session to encrypt the security data and in case of old session is leaked[3], confidentiality is not preserved any more. A possible attack is shown below where $M$ represents the attacker:

$$\begin{aligned}
1.\ & [A]_1 \rightarrow [S]_1 : \{[B]_1, [K_{ab}]_1\}_{[K_A]_1} \\
2.\ & [S]_1 \rightarrow M : \{[A]_1, [K_{ab}]_1\}_{[K_B]_1} \\
& M \rightarrow [B]_1 : \{[A]_0, [K_{ab}]_0\}_{[K_B]_0} \\
3.\ & [B]_1 \rightarrow [A]_1 : \{[Msg]_1\}_{[K_{ab}]_0}
\end{aligned}$$

### 3.4 Semantic properties

We prove below that our analysis respects the operational semantics of extended LySA. More precisely, we prove a subject reduction result for both the standard and the reference monitor semantics: if $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$, then the same triple

[3] This is possible because the attacker normally has unlimited time and resource to break the code

$(\rho, \kappa, \psi)$ is a valid estimate for all the states passed through in a computation of $\mathcal{P}$, i.e. for all the derivatives of $\mathcal{P}$. Additionally, we show that when the $\psi$ component is empty, then the reference monitor is useless.

It is convenient to prove the following lemmata. The first states that estimates are resistant to substitution of closed terms for variables, and it holds for both extended terms and processes. The second lemma says that an estimate for an extended processes $\mathcal{P}$ is valid for every process congruent to $\mathcal{P}$, as well.

**Lemma 1.** ***(Substitution result)***

1. $\rho \models \mathcal{E} : \vartheta$ *and* $\mathcal{E}' \in \rho(x)$ *imply* $\rho \models \mathcal{E}[\mathcal{E}'/x] : \vartheta$
2. $\rho, \kappa \models_{\mathsf{RM}} P : \psi$ *and* $\mathcal{E} \in \rho(x)$ *imply* $\rho, \kappa \models_{\mathsf{RM}} P[\mathcal{E}/x] : \psi$

*Proof of 1* The proof proceeds by structural induction over expression by regarding each of the rules in the analysis.

**Case (Name)**. Assume that $\rho \models [n]_s : \vartheta$. For arbitrary choices of $\mathcal{E}'$ and $x$ it holds that $[n]_s[\mathcal{E}'/x] = [n]_s$ so it is immediate that also $\rho \models [n]_s[\mathcal{E}'/x] : \vartheta$.

**Case (Variable)**. Assume that $\mathcal{E} = x'$, i.e. that $\rho \models x' : \vartheta$ and therefore $\rho(x') \subseteq \vartheta$. Then there are two cases. Either $\mathcal{E} \neq x$ in which case $x'[\mathcal{E}'/x] = x'$ so clearly $\rho \models x'[\mathcal{E}'/x] : \vartheta$. Alternatively, $\mathcal{E} = x$ in which case $x'[\mathcal{E}'/x] = \mathcal{E}'$. Furthermore assume that $\mathcal{E}' \in \rho(x)$ and because $\rho(x') \subseteq \vartheta$, it holds that $\rho \models \mathcal{E}' : \vartheta$ in which case $\rho \models \mathcal{E}[\mathcal{E}'/x] : \vartheta$ by the analysis.

**Case (Encryption)**. Assume that $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}$, i.e. $\rho \models \{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0} : \vartheta$. The result holds by applying the induction hypothesis on each individual $\mathcal{E}_i$.

*Proof of 2* The proof is done by straightforward induction applying the induction hypothesis on any sub-process and lemma 1.1 on any sub-terms.

**Lemma 2.** ***(Congruence)***
*If* $P \equiv Q$ *and* $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) : \psi$ *then* $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([Q]_s) : \psi$

*Proof* The proof amounts to a straightforward inspection of each of the clauses defining $P \equiv Q$, e.g.
**In case** $P \mid 0 \equiv P$
We assume $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P \mid 0]_s) : \psi$, which amounts to $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) \mid 0 : \psi$ according to the definition of $\mathcal{T}$, then it must be the case that

$$\frac{\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) : \psi \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} 0 : \psi}{\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) \mid 0 : \psi}$$

by the analysis rule. Therefore $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) : \psi$ is proved.
**Other cases** are similar and we skip the proofs because of space.

We are now ready to state the subject reduction result. It expresses that our analysis is semantically correct regardless of the way the semantics is parameterised, furthermore the reference monitor can not abort $\mathcal{P}$ when $\psi$ is empty.

**Theorem 1.** ***(Subject reduction)***

1. *If* $\mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{Q}$ *and* $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$ *then also* $\rho, \kappa \models_{\mathsf{RM}} \mathcal{Q} : \psi$;
2. *Furthermore, if* $\psi = \emptyset$ *then* $\mathcal{P} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{Q}$

*Proof* By induction on the inference of $\mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{Q}$.
**In case (Com)** we assume
$\rho, \kappa \models_{\mathsf{RM}} \langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle.\mathcal{P} \mid (\mathcal{E}'_1, \ldots, \mathcal{E}'_j; x_{j+1}, \ldots, x_k).\mathcal{Q} : \psi$ which amounts to:
$(a)$ $\wedge_{i=1}^{k} \rho \models \mathcal{E}_i : \vartheta_i$
$(b)$ $\forall v_1, \ldots, v_k : \wedge_{i=1}^{k} v_i \in \vartheta_i \Rightarrow \langle v_1, \ldots, v_k \rangle \in \kappa$
$(c)$ $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$
$(d)$ $\wedge_{i=1}^{j} \rho \models \mathcal{E}'_i : \vartheta'_i$
$(e)$ $\forall \langle v_1, \ldots, v_k \rangle \in \kappa : \wedge_{i=1}^{j} v_i \propto \vartheta'_i \Rightarrow \wedge_{i=j+1}^{k} v_i \in \rho(x_i) \ \wedge \ \rho, \kappa \models_{\mathsf{RM}} \mathcal{Q} : \psi$
Moreover we assume that $\wedge_{i=1}^{j} \mathcal{E}_i \approx \mathcal{E}'_i$ because
$\langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle.\mathcal{P} \mid (\mathcal{E}'_1, \ldots, \mathcal{E}'_j; x_{j+1}, \ldots, x_k).\mathcal{Q} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{P} \mid \mathcal{Q}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k]$
and we have to prove $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} \mid \mathcal{Q}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k] : \psi$. From $(a)$ we have $\wedge_{i=1}^{k} \mathcal{E}_i \in \vartheta_i$ since $\wedge_{i=1}^{k} \mathsf{fv}(\mathcal{E}_i) = \emptyset$ and then $(b)$ gives $\langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle \in \kappa$. From $(d)$ and the assumption $\wedge_{i=1}^{j} \mathcal{E}_i \approx \mathcal{E}'_i$ we get $\wedge_{i=1}^{j} \mathcal{E}_i \propto \vartheta'_i$. Now $(e)$ gives $\wedge_{i=j+1}^{k} \mathcal{E}_i \in \rho(x_i)$ and $\rho, \kappa \models_{\mathsf{RM}} \mathcal{Q} : \psi$. The substitution result (Lemma 1) then gives $\rho, \kappa \models_{\mathsf{RM}} \mathcal{Q}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k] : \psi$ and together with $(c)$ this gives the required result.
The second part is trivial: when $\psi = \emptyset$, obviously
$\langle \mathcal{E}_1, \ldots, \mathcal{E}_k \rangle.\mathcal{P} \mid (\mathcal{E}'_1, \ldots, \mathcal{E}'_j; x_{j+1}, \ldots, x_k).\mathcal{Q} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{P} \mid \mathcal{Q}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k]$

**In case (Decr)** we assume
$\rho, \kappa \models_{\mathsf{RM}} \mathit{decrypt}\ [\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s\ \mathit{as}\ \{\mathcal{E}'_1, \ldots, \mathcal{E}'_j; x_{j+1}, \ldots, x_k\}_{\mathcal{E}'_0}\ \mathit{in}\ \mathcal{P} : \psi$
which amounts to:
$(f)$ $\wedge_{i=0}^{k} \ \rho \models \mathcal{E}_i : \vartheta_i$
$(g)$ $\forall v_0, \ldots, v_k : \wedge_{i=0}^{k} v_i \in \vartheta_i \Rightarrow [\{v_1, \ldots, v_k\}_{v_0}]_s \in \vartheta$
$(h)$ $\wedge_{i=0}^{j} \rho \models \mathcal{E}'_i : \vartheta'_i$
$(i)$ $\forall [\{v_1, \ldots, v_k\}_{v_0}]_{s'} \in \vartheta : \wedge_{i=0}^{j} v_i \propto \vartheta'_i$
$\Rightarrow \wedge_{i=j+1}^{k} v_i \in \rho(x_i) \ \wedge$
$\wedge_{i=0}^{j} \neg RM((\Gamma(\mathcal{E}_0), \Gamma(\mathcal{E}'_0)), \ldots, (\Gamma(\mathcal{E}_j), \Gamma(\mathcal{E}'_j))) \Rightarrow (x_{j+1}, \ldots, x_k) \in \psi \ \wedge$
$\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$
Furthermore we assume that $\wedge_{i=0}^{j} \mathcal{E}_i \approx \mathcal{E}'_i$ because
$\mathit{decrypt}\ [\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s\ \mathit{as}\ \{\mathcal{E}'_1, \ldots, \mathcal{E}'_j; x_{j+1}, \ldots, x_k\}_{\mathcal{E}'_0}\ \mathit{in}\ \mathcal{P} \xrightarrow{\alpha}_{\mathcal{R}}$
$\mathcal{P}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k]$ and we have to prove $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k] : \psi$. From $(f)$ and $\wedge_{i=0}^{k} \mathsf{fv}(\mathcal{E}_i) = \emptyset$, we get $\wedge_{i=0}^{k} \mathcal{E}_i \in \vartheta_i$ and then $(g)$ gives $[\{\mathcal{E}_1, \ldots, \mathcal{E}_k\}_{\mathcal{E}_0}]_s \in \vartheta$. From $(h)$ and the assumption $\wedge_{i=0}^{j} \mathcal{E}_i \approx \mathcal{E}'_i$ we get $\wedge_{i=0}^{j} \mathcal{E}_i \propto \vartheta'_i$. Now $(i)$ gives $\wedge_{i=j+1}^{k} \mathcal{E}_i \in \rho(x_i)$ and $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \psi$. Using Lemma 1 we get the required result $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P}[\mathcal{E}_{j+1}/x_{j+1}, \ldots, \mathcal{E}_k/x_k] : \psi$
For the second part of the result we observe that
$\neg RM((\Gamma(\mathcal{E}_0), \Gamma(\mathcal{E}'_0)), \ldots, (\Gamma(\mathcal{E}_j), \Gamma(\mathcal{E}'_j))) \Rightarrow (x_{j+1}, \ldots, x_k) \in \psi$ follows from $(i)$ and since $\psi = \emptyset$ it must be the case that $RM((\mathcal{I}(\mathcal{E}_0), \mathcal{I}(\mathcal{E}'_0)), \ldots, (\mathcal{I}(\mathcal{E}_j), \mathcal{I}(\mathcal{E}'_j)))$. Thus the condition of the rule (Decr) are fulfilled for $\xrightarrow{\alpha}_{\mathsf{RM}}$.

**In case (Repl)** we assume
$\rho, \kappa \models_{\mathsf{RM}} [!P]_s : \psi$, which amounts to:
$(a)$ $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_s) : \psi$
$(b)$ $\rho, \kappa \models_{\mathsf{RM}} \mathcal{T}([P]_{s'}) : \psi$
$(c)$ $\rho, \kappa \models_{\mathsf{RM}} [!P]_{s'} : \psi$ (because $\psi$ does not contain any session information)
$(a)$ together with $(c)$ then gives the required result $\rho, \kappa \models_{\mathsf{RM}} (\mathcal{T}([P]_s) \mid [!P]_{s'}) : \psi$.
Furthermore, it is obviously that when $\psi = \emptyset$, $[!P]_s \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{T}([P]_s) \mid [!P]_{s'}$

The cases (Par) and (Res) follow directly from the induction hypothesis. The case (Congr) also uses the congruence result.

The next result shows that our analysis correctly predicts when we can safely dispense with the reference monitor. We shall say that the reference monitor RM *cannot abort* a process $\mathcal{P}$ when there exist no $\mathcal{Q}, \mathcal{Q}'$ such that $\mathcal{P} \xrightarrow{\alpha}{}^*_{\mathcal{R}} \mathcal{Q} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{Q}'$ and $\mathcal{P} \xrightarrow{\alpha}{}^*_{\mathsf{RM}} \mathcal{Q} \not\rightarrow_{\mathsf{RM}}$. As usual, * stands for the transitive and reflexive closure of the relation in question, and $\mathcal{Q} \not\rightarrow_{\mathsf{RM}}$ stands for $\nexists \mathcal{Q}' : \mathcal{Q} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{Q}'$. We then have:

**Theorem 2.** ***(Static check for reference monitor)***

- *If $\rho, \kappa \models_{\mathsf{RM}} \mathcal{P} : \emptyset$ then RM cannot abort $\mathcal{P}$.*

*Proof* Suppose *per absurdum* that such $\mathcal{Q}$ and $\mathcal{Q}'$ exist. A straightforward induction extends the subject reduction result to $\mathcal{P} \xrightarrow{\alpha}{}^*_{\mathcal{R}} \mathcal{Q}$ giving $\rho, \kappa \models_{\mathsf{RM}} \mathcal{Q} : \emptyset$. The part 2 of the subject reduction result applied to $\mathcal{Q} \xrightarrow{\alpha}_{\mathcal{R}} \mathcal{Q}'$ gives $\mathcal{Q} \xrightarrow{\alpha}_{\mathsf{RM}} \mathcal{Q}'$ which is a contradiction.

## 4 Conclusion

In this paper we have introduced a sound way to detect replay attacks. To do that, we extended the standard LySA calculus with session identifiers and gave it a reduction semantics. Unlike the original LySA semantics [2, 3], the reduction relation carries labels on the arrows, which gives the ability to faithfully describe what happens during the process evolution at a certain degree of accuracy, and facilities the reasoning about dynamic properties.

On the static side, we developed a control flow analysis to verify the freshness property of the extended processes. The static property ensures that, if the secret information received by a principal is in the right context, then a process is not subject to a run-external attack at execution time. It is not difficult to incorporate other relevant techniques, e.g. the one from [6], into our framework in order to take care of run-internal attacks as well.

We implemented the analysis and used our tool to check some significant protocols, e.g. Wide Mouthed Frog, Yahalom, Andrew Secure RPC, Otway-Rees, Needham-Schroeder, Amended Needham-Schroeder. The tool confirmed that we can successfully detect potential replay attacks on the protocols.

## References


1. Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. Information and Computation, 148(1), pp. 1-70, 1999.
2. Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson and Hanne Riis Nielson. Automatic Valication of Protocol Narration. In Proc. of Computer Security Foundations Workshop XVI, IEEE Press, pp. 126-140, 2003.
3. Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson and Hanne Riis Nielson. Static Validation of Security Protocols. Journal of Computer Security, 13(3), pp.347 - 390, 2005.
4. Michael Burrows and Martín Abadi and Roger Needham. A Logic of Authentication, ACM. Transactions in Computer Systems, 8(1), pp. 18-36, 1990.
5. Michele Curti, Pierpaolo Degano and Cosima Tatiana Baldari. Causal $\pi$-Calculus for Biochemical Modelling. In Proc. of Workshop on Computational Methods in Systems Biology, LNCS 2602, pp. 21-33, 2003.
6. Han Gao and Hanne Riis Nielson. Analysis of LYSA-calculus with explicit confidentiality annotations. 20th International Conference on Advanced Information Networking and Applications (AINA 2006), IEEE Computer Society.
7. Catherine Meadows, Paul Syverson, and Iliano Cervesato. Formal Specification and Analysis of the Group Domain of Interpretation Protocol Using NPATRL and the NRL Protocol Analyzer. Journal of Computer Security. 12(6), pp. 893-931, 2004.
8. Jonathan K. Millen. Term Replacement Algebra for the Interrogator. The MITRE Corporation, MP 97B65, 1997.
9. Robin Milner. Communicating and mobile systems: the $\pi$-calculus. Cambridge University Press, 1999.
10. Roger Needham and Michael Schroeder. Using encryption for authentication in large networks of computers. Communications of the ACM, 21(12), Dec 1978.
11. Dave Otway and Owen Rees. Efficient and Timely Mutual Authentication. Operating Systems Review, 21(1), pp.8-10, 1987.
12. Lawrence C. Paulson. Inductive Analysis of the Internet Protocol TLS. ACM Transactions on Computer and System Security, 1999, 2(3): 332-351.
13. Lawrence C Paulson. The foundation of a generic theorem prover. Automated Reasoning 5 (1989), pp. 363-397.
14. Paul Syverson. A Taxonomy of Replay attacks. In Proc. of Computer Security Foundations Workshop VII, IEEE Computer Society Press, 1994.