Non-Repudiation Analysis Using LySa with Annotations

Mayla Brusò and Agostino Cortesi

Abstract This work introduces a formal analysis of the non-repudiation property for security protocols. Protocols are modelled in the process calculus LySa, using an extended syntax with annotations. Non-repudiation is verified using a Control Flow Analysis, following the same approach of M. Buchholtz and H. Gao for authentication and freshness analyses.
The result is an analysis that can statically check the protocols to predict if they are secure during their execution and which can be fully automated.

1 Introduction

With the growth of Internet applications like e-shopping or e-voting, non-repudiation is becoming increasingly important, as a protocol property. Our aim is to provide a protocol analysis which checks this property to avoid that a protocol is used in malicious way. Among the existing techniques that perform the analysis of non-repudiation protocols, we may cite:

- The CSP (Communicating Sequential Processes) approach [10], [11]: it is an abstract language designed specifically for the description of communication patterns of concurrent system components that interact through message passing.
- The game approach [8]: it considers the execution of the protocol as a game, where each entity is a player; the protocols are designed finding a strategy, which has to defend an honest entity against all the possible strategies of malicious parties.

Mayla Brusò
Computer Science Department, Ca' Foscari University, Italy, e-mail: mabruso@dsi.unive.it

Agostino Cortesi
Computer Science Department, Ca' Foscari University, Italy, e-mail: acortesi@dsi.unive.it

- The Zhou-Gollmann approach [14]: it uses *SVO Logic*, a modal logic that is composed by inference rules and axioms which are used to express beliefs that can be analysed by a judge to decide if the service provided the property.
- The inductive approach [1]: it uses an inductive model, a set of all the possible histories of the network that the protocol execution may produce; a history, called *trace*, is a list of network events, that can indicate the communication of a message or the annotation of information for future use.

We follow a different approach, the same as M. Buchholtz [3] and H. Gao [6], who show how some security properties can be analysed using the LYSA [2] process calculus with annotations and a Control Flow Analysis (CFA) to detect flaws in the protocols. The main idea is to extend LYSA with specific annotations, i.e. tags that identify part of the message for which the property has to hold and that uniquely assign principal and session identifiers to encryptions and decryptions.

It is interesting to notice that the non-repudiation analysis that we propose easily fits into the CFA framework [9], yielding a suite of analyses that can be combined in various ways, with no major implementation overload.

The main differences between our proposal and the previously cited alternative approaches are the following: our analysis can check many protocols and can model scenarios with infinitely many principals while other approaches often are developed to analyse only a particular protocol and can model scenarios with finite principals. Moreover LYSA does not consider channels, like other approaches does, therefore a more realistic environment in which the messages are not guaranteed to reach the destination is not considered.

The structure of the paper is the following: Section 2 is a quick overview of LYSA; Section 3 presents the CFA framework; Section 4 shows the new non-repudiation analysis, and its application to the protocols; Section 5 concludes.

2 LYSA

LYSA [2] is a process calculus in the π-calculus tradition that models security protocols on a global network. It incorporates pattern matching into the language constructs where values can become bound to variables. In LYSA all the communications take place directly on a global network and this corresponds to the scenario in which security protocols often operate. Channels are considered in many process calculi, but they may give a degree of security that there is not in the common network, where a spy can eavesdrop and forge communications; furthermore, private channels are often used explicitly as cryptographic keys. LYSA calculus offers a realistic environment in which there are not channels to protect the exchange of messages among the principals.

2.1 Syntax and Semantics

An expression $E \in Expr$ may represent a name, a variable or an encryption. The set *Expr* contains two disjointed subsets, *Name* and *Var*. The elements in the first subset can be identifiers, nonces, symmetric keys or key pairs m^+ and m^- for asymmetric key cryptography, etc., ranged over by n. The elements in *Var* are only variables, ranged over by x. The remaining expressions are symmetric and asymmetric encryptions of k-tuples of other expressions, defined as $\{E_1,\dots,E_k\}_{E_0}$ and $\{|\, E_1,\dots,E_k \,|\}_{E_0}$ respectively, where E_0 represents a symmetric or asymmetric key.

LYSA also allows to construct processes $P \in Proc$, which use the expressions explained above. Processes can have the following form:

- $\langle E_1,\dots,E_k\rangle.P$: the process sends a k-tuple of values onto the global network; if the message reaches its destination, the process continues as P.
- $(E_1,\dots,E_j;x_{j+1},\dots,x_k).P$: the process reads the k-tuple of values sent, it checks if the values expected are identical to $E_1,\dots,E_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $x_{j+1},\dots,x_k$, and the process continues as P, which is the scope of the variables; notice that a semi-colon is used to distinguish between the expressions used for matching and the variables.
- decrypt E as $\{E_1,\dots,E_j;x_{j+1},\dots,x_k\}_{E_0}$ in P: the process denotes the symmetric decryption and it works in a way similar to the input construct; if the encryption key is identical to E_0, the process decrypts the k-tuple, then it checks if the values expected are identical to $E_1,\dots,E_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $x_{j+1},\dots,x_k$, and the process continues as P, which is the scope of the variables; a semi-colon distinguishes between the expressions used for matching and the variables.
- decrypt E as $\{|\, E_1,\dots,E_j;x_{j+1},\dots,x_k \,|\}_{E_0}$ in P: the process denotes the asymmetric decryption and it works like symmetric decryption; the only differences are in E_0 and in the key used to encrypt, which have to be a key pair m^+ and m^-; their order depends on the role of decryption, i.e. if it is used to verify a private key signature or to obtain the original message after a public key encryption.
- $(\nu\, n)P$: the process generates a new name n and it continues in P, which is the scope of the name.
- $(\nu \pm m)P$: the process generates a new key pair, m^+ and m^-, and it continues in P, which is the scope of the key pair.
- $P_1 \mid P_2$: the process denotes two processes running in parallel that may synchronize through communication over the network or perform actions independently.
- $!P$: the process acts as an arbitrary number of processes P composed in parallel.
- 0: the process is the inactive or nil process that does nothing.

Both expressions and processes are defined in the Table 1.

A binder introduces new names or variables which have scope in the rest of the process. The prefix $(\nu\, n)$ in the process $(\nu\, n)P$ and the prefix $(\nu \pm m)$ in the process $(\nu \pm m)P$ are binders, because they create new keys which have scope in the process P. Also input and decryption are binders that introduce the variables $x_{j+1},\dots,x_k$. If a name or a variable is not bound by any binder, it is *free*; the function $fn(P)$ collects all the free names in the process P and it is defined in the Table 2 while the function $fv(P)$, defined in the Table 3, collects the free variables. The bound variables are defined by the function $bv(P) \stackrel{def}{=} var(P) \setminus fv(P)$, i.e. they are all the variables that are not free. All these functions are also defined on the terms, which are part of the processes.

Table 1 Syntax of LYSA calculus

$E ::=$	*terms*
n	name
x	variable
m^+	public key
m^-	private key
$\{E_1,\ldots,E_k\}_{E_0}$	symmetric encryption
$\{\mid E_1,\ldots,E_k \mid\}_{E_0}$	asymmetric encryption
$P ::=$	*processes*
$\langle E_1,\ldots,E_k\rangle.P$	output
$(E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P$	input
decrypt E as $\{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0}$ in P	symmetric decryption
decrypt E as $\{\mid E_1,\ldots,E_j;x_{j+1},\ldots,x_k \mid\}_{E_0}$ in P	asymmetric decryption
$(\nu\, n)P$	restriction
$(\nu \pm m)P$	pair restriction
$P_1 \mid P_2$	parallel composition
$!P$	replication
0	nil

LYSA provides a reduction semantics that describes the evolution of a process step-by-step, using a *reduction relation* between two processes, written $P \rightarrow P'$. If the reduction relation holds then P can evolve in P' using the rules depicted in Table 6, that show an inductive definition of the relation by axioms and inference rules.

The structural congruence between two processes, written $P \equiv P'$, means that P is equal to P' except for syntactic aspects, but this does not interfere with the way they evolve. The structural congruence is defined as the smallest relation satisfying the rules in the Table 4, that express the following ideas:

- The reduction relation is an equivalence relation.
- The parallel composition is defined to be commutative, associative, and has 0 as neutral element.
- The order of the processes in the parallel composition is not influential.
- The replication corresponds to an arbitrary number of process in parallel.
- The restrictions can be simplified under certain assumptions.
- Two processes are structurally equivalent whenever they are α-equivalent.

Two processes P_1 and P_2 are α-equivalent, written $P_1 \stackrel{\alpha}{\equiv} P_2$, when they are identical except that they may differ in the choice of bound names. A procedure called α-*conversion* replaces all the instances of a bound name in a process for another

Table 2 Function $fn(P)$ for free names

$fn(n)$	$\stackrel{def}{=} \{n\}$
$fn(m^+)$	$\stackrel{def}{=} \{m^+\}$
$fn(m^-)$	$\stackrel{def}{=} \{m^-\}$
$fn(x)$	$\stackrel{def}{=} \emptyset$
$fn(\{E_1,\ldots,E_k\}_{E_0})$	$\stackrel{def}{=} fn(E_0) \cup \ldots \cup fn(E_k)$
$fn(\{\mid E_1,\ldots,E_k \mid\}_{E_0})$	$\stackrel{def}{=} fn(E_0) \cup \ldots \cup fn(E_k)$
$fn(\langle E_1,\ldots,E_k\rangle.P)$	$\stackrel{def}{=} fn(E_1) \cup \ldots \cup fn(E_k) \cup fn(P)$
$fn((E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P)$	$\stackrel{def}{=} fn(E_1) \cup \ldots \cup fn(E_j) \cup fn(P)$
$fn(\text{decrypt } E \text{ as } \{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0} \text{ in } P)$	$\stackrel{def}{=} fn(E) \cup fn(E_0) \cup \ldots \cup fn(E_j) \cup fn(P)$
$fn(\text{decrypt } E \text{ as } \{\mid E_1,\ldots,E_j;x_{j+1},\ldots,x_k \mid\}_{E_0} \text{ in } P)$	$\stackrel{def}{=} fn(E) \cup fn(E_0) \cup \ldots \cup fn(E_j) \cup fn(P)$
$fn((\nu\, n)P)$	$\stackrel{def}{=} fn(P) \setminus \{n\}$
$fn((\nu \pm m)P)$	$\stackrel{def}{=} fn(P) \setminus \{m^+, m^-\}$
$fn(P_1 \mid P_2)$	$\stackrel{def}{=} fn(P_1) \cup fn(P_2)$
$fn(!P)$	$\stackrel{def}{=} fn(P)$
$fn(0)$	$\stackrel{def}{=} \emptyset$

Table 3 Function $fv(P)$ for free variables

$fv(n)$	$\stackrel{def}{=} \emptyset$
$fv(m^+)$	$\stackrel{def}{=} \emptyset$
$fv(m^-)$	$\stackrel{def}{=} \emptyset$
$fv(x)$	$\stackrel{def}{=} \{x\}$
$fv(\{E_1,\ldots,E_k\}_{E_0})$	$\stackrel{def}{=} fv(E_0) \cup \ldots \cup fv(E_k)$
$fv(\{\mid E_1,\ldots,E_k \mid\}_{E_0})$	$\stackrel{def}{=} fv(E_0) \cup \ldots \cup fv(E_k)$
$fv(\langle E_1,\ldots,E_k\rangle.P)$	$\stackrel{def}{=} fv(E_1) \cup \ldots \cup fv(E_k) \cup fv(P)$
$fv((E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P)$	$\stackrel{def}{=} fv(E_1) \cup \ldots \cup fv(E_j) \cup (fv(P) \setminus \{x_{j+1},\ldots,x_k\})$
$fv(\text{decrypt } E \text{ as } \{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0} \text{ in } P)$	
	$\stackrel{def}{=} fv(E_0) \cup \ldots \cup fv(E_j) \cup (fv(P) \setminus \{x_{j+1},\ldots,x_k\})$
$fv(\text{decrypt } E \text{ as } \{\mid E_1,\ldots,E_j;x_{j+1},\ldots,x_k \mid\}_{E_0} \text{ in } P)$	
	$\stackrel{def}{=} fv(E_0) \cup \ldots \cup fv(E_j) \cup (fv(P) \setminus \{x_{j+1},\ldots,x_k\})$
$fv((\nu\, n)P)$	$\stackrel{def}{=} fv(P)$
$fv((\nu \pm m)P)$	$\stackrel{def}{=} fv(P)$
$fv(P_1 \mid P_2)$	$\stackrel{def}{=} fv(P_1) \cup fv(P_2)$
$fv(!P)$	$\stackrel{def}{=} fv(P)$
$fv(0)$	$\stackrel{def}{=} \emptyset$

Table 4 Structural congruence $P \equiv P'$

$$
\begin{array}{l}
P \equiv P \\
P_1 \equiv P_2 \Rightarrow P_2 \equiv P_1 \\
P_1 \equiv P_2 \wedge P_2 \equiv P_3 \Rightarrow P_1 \equiv P_3 \\
P_1 \equiv P_2 \Rightarrow \begin{cases}
\langle E_1, \dots, E_k \rangle.P_1 \equiv \langle E_1, \dots, E_k \rangle.P_2 \\
(E_1, \dots, E_j; x_{j+1}, \dots, x_k).P_1 \equiv \\
\quad (E_1, \dots, E_j; x_{j+1}, \dots, x_k).P_2 \\
\text{decrypt } E \text{ as } \{E_1, \dots, E_j; x_{j+1}, \dots, x_k\}_{E_0} \text{ in } P_1 \equiv \\
\quad \text{decrypt } E \text{ as } \{E_1, \dots, E_j; x_{j+1}, \dots, x_k\}_{E_0} \text{ in } P_2 \\
\text{decrypt } E \text{ as } \{\mid E_1, \dots, E_j; x_{j+1}, \dots, x_k \mid\}_{E_0} \text{ in } P_1 \equiv \\
\quad \text{decrypt } E \text{ as } \{\mid E_1, \dots, E_j; x_{j+1}, \dots, x_k \mid\}_{E_0} \text{ in } P_2 \\
(\nu\, n)P_1 \equiv (\nu\, n)P_2 \\
(\nu \pm m)P_1 \equiv (\nu \pm m)P_2 \\
P_1 \mid P_3 \equiv P_2 \mid P_3 \\
!P_1 \equiv !P_2
\end{cases} \\
P_1 \mid P_2 \equiv P_2 \mid P_1 \\
(P_1 \mid P_2) \mid P_3 \equiv P_1 \mid (P_2 \mid P_3) \\
P \mid 0 \equiv P \\
!P \equiv P \mid !P \\
(\nu\, n)0 \equiv 0 \\
(\nu\, n_1)(\nu\, n_2)P \equiv (\nu\, n_2)(\nu\, n_1)P \\
(\nu\, n)(P_1 \mid P_2) \equiv P_1 \mid (\nu\, n)P_2 \quad \text{if } n \notin fn(P_1) \\
(\nu \pm m)0 \equiv 0 \\
(\nu \pm m_1)(\nu \pm m_2)P \equiv (\nu \pm m_2)(\nu \pm m_1)P \\
(\nu \pm m)(P_1 \mid P_2) \equiv P_1 \mid (\nu \pm m)P_2 \quad \text{if } m^+, m^- \notin fn(P_1) \\
(\nu \pm m)(\nu\, n)P \equiv (\nu\, n)(\nu \pm m)P \\
P_1 \overset{\alpha}{\equiv} P_2 \Rightarrow P_1 \equiv P_2
\end{array}
$$

name. The definition of the equivalence relation is in the Table 5. Notice that a substitution $P[n_1 \mapsto n_2]$ substitutes all the free occurrences of n_1 in P for n_2.

Table 5 α-equivalence $\overset{\alpha}{\equiv}$

$$
\begin{array}{l}
P \overset{\alpha}{\equiv} P \\
P_1 \overset{\alpha}{\equiv} P_2 \text{ implies } P_2 \overset{\alpha}{\equiv} P_1 \\
P_1 \overset{\alpha}{\equiv} P_2 \wedge P_2 \overset{\alpha}{\equiv} P_3 \text{ implies } P_1 \overset{\alpha}{\equiv} P_3 \\
(\nu\, n_1)P \overset{\alpha}{\equiv} (\nu\, n_2)(P[n_1 \mapsto n_2]) \quad \text{if } n_2 \notin fn(P) \\
(\nu \pm m_1)P \overset{\alpha}{\equiv} (\nu \pm m_2)(P[m_1^+ \mapsto m_2^+, m_1^- \mapsto m_2^-]) \quad \text{if } m_2^+, m_2^- \notin fn(P)
\end{array}
$$

Finally, we define values $V \in Val$, which are used in the reduction as expressions without variables $x \in Var$:

$$
\begin{aligned}
V ::= \;& n \\
& \mid m^{+} \\
& \mid m^{-} \\
& \mid \{V_1, \ldots, V_k\}_{V_0} \\
& \mid \{\!|\, V_1, \ldots, V_k \,|\!\}_{V_0}
\end{aligned}
$$

The reduction relation describes how a process may evolve into another and it is defined inductively as the smallest relation such that the rules in the Table 6 are satisfied. A *reference monitor* is used to check each step before allowing it to be executed. It can be turned off or on: in the first case there are not requirements that have to be meet; in the other case some properties are checked at run time and, if the check does not succeed, the process execution is aborted.

A *substitution function* is used in the reduction rules, written $P[V/x]$; it substitutes a variable x for a value V in the process P whenever x becomes bound to V.

The rule (Com) is the parallel composition between an output process and an input process. This means that the communication between two principals happens only if these two processes run in parallel. Furthermore, the first j values $V_1, \ldots, V_j$ sent have to be identical to the first j values $V'_1, \ldots, V'_j$ that the recipient expects. In this case, the variables are substituted with the values $V_{j+1}, \ldots, V_k$. The rules (Dec), (ADec) and (ASig) are used to decrypt messages with a symmetric key, a private key and a public key respectively. As before, the first j values $V_1, \ldots, V_j$ encrypted have to be identical to the first j values $V'_1, \ldots, V'_j$ that who decrypts the message expects. In this case, the variables are substituted with the values $V_{j+1}, \ldots, V_k$. The rule (New) and (ANew) restrict the scope of the names created, therefore they are visible only in the respective process. The rule (Par) is the parallel composition that can evolve in a new parallel composition where one of the two processes involved is evolved while the other remains unchanged. The rule (Congr) allows to apply the reduction relation to any process that is structurally congruent to the process found in the other rules.

2.2 Meta Level Calculus

The meta level is an extension of LySA that can be used to describe different scenarios in which many principals execute a protocol at the same time. Thanks to this level the analysis can run in a realistic environment with many initiators and responders. This is done by running several copies of the processes and renaming each name and each variable using indexes, added to make them unique.

The syntax of the meta level is defined by the grammar described in Table 7. Its constructs incorporate a countable indexing set S, which includes a set of variables X.

Table 6 Semantics of LYSA calculus

(Com) $$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{\langle V_1,\ldots,V_k\rangle.P \mid (V_1',\ldots,V_j';x_{j+1},\ldots,x_k).P' \to_{\mathscr{R}} P \mid P'[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$$

(Dec) $$\frac{\bigwedge_{i=0}^{j} V_i = V_i'}{\text{decrypt } \{V_1,\ldots,V_k\}_{V_0} \text{ as } \{V_1',\ldots,V_j';x_{j+1},\ldots,x_k\}_{V_0'} \text{ in } P \to_{\mathscr{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$$

(ADec) $$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{\text{decrypt } \{\mid V_1,\ldots,V_k \mid\}_{m^+} \text{ as } \{\mid V_1',\ldots,V_j';x_{j+1},\ldots,x_k \mid\}_{m^-} \text{ in } P \to_{\mathscr{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$$

(ASig) $$\frac{\bigwedge_{i=1}^{j} V_i = V_i'}{\text{decrypt } \{\mid V_1,\ldots,V_k \mid\}_{m^-} \text{ as } \{\mid V_1',\ldots,V_j';x_{j+1},\ldots,x_k \mid\}_{m^+} \text{ in } P \to_{\mathscr{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$$

(New) $$\frac{P \to_{\mathscr{R}} P'}{(\nu\, n)P \to_{\mathscr{R}} (\nu\, n)P'}$$

(ANew) $$\frac{P \to_{\mathscr{R}} P'}{(\nu \pm m)P \to_{\mathscr{R}} (\nu \pm m)P'}$$

(Par) $$\frac{P_1 \to_{\mathscr{R}} P_1'}{P_1 \mid P_2 \to_{\mathscr{R}} P_1' \mid P_2}$$

(Congr) $$\frac{P \equiv P' \,\wedge\, P' \to_{\mathscr{R}} P'' \,\wedge\, P'' \equiv P'''}{P \to_{\mathscr{R}} P'''}$$

The meta level terms are identical to the object level terms, i.e. the terms explained before, except that names, variables and asymmetric keys are indexed. A sequence of indexes $\bar{i}$ is added as subscript, that is a shorthand for $i_1,\ldots,i_k$. The meta level processes are the following:

- $|_{i \in S}\, MP$: the process describes the parallel composition of instances of the process MP where the index i is an element in the set S.
- let $X \subseteq S$ in MP: the process declares a set identifier X which has some values of the index set S in the process MP; the set X can be infinite.
- $(\nu_{\bar{i} \in \bar{S}}\, n_{\overline{ai}})MP$: the process describes the restriction of all the names $n_{\overline{ai}}$; $\overline{a}$ is a prefix of the index that can be empty.
- $(\nu_{\pm \bar{i} \in \bar{S}}\, m_{\overline{ai}})MP$: the process describes the restriction of all the key pairs $m^{+}_{\overline{ai}}$ and $m^{-}_{\overline{ai}}$; as above, $\overline{a}$ is a prefix of the index that can be empty.
- $\langle ME_1,\ldots,ME_k\rangle.MP$: the process sends a k-tuple of values onto the global network; if the message reaches its destination, the process continues as MP.
- $(ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k).MP$: the process reads the k-tuple of values sent, it checks if the values expected are identical to $ME_1,\ldots,ME_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $mx_{j+1},\ldots,mx_k$, and the process continues as MP, which is the scope of the variables; a semi-colon is used to distinguish between the terms used for matching and the variables, as in the input process seen in the object level.
- decrypt ME as $\{ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k\}_{ME_0}$ in MP: the process denotes the symmetric decryption; it checks if the encryption key is identical to E_0, then

the process decrypts the k-tuple, and it checks if the values expected are identical to $ME_1,\dots,ME_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $mx_{j+1},\dots,mx_k$, and the process continues as MP, which is the scope of the variables.
- decrypt ME as $\{\!|\ ME_1,\dots,ME_j;mx_{j+1},\dots,mx_k\ |\!\}_{ME_0}$ in MP: the process denotes the asymmetric decryption and it works like symmetric decryption except that E_0 and the key used to encrypt have to be a key pair m^+ and m^-.
- $(\nu\ n_{\bar{i}})MP$: the process generates k new names n_i, $i \in [1..k]$, and it continues in MP, which is the scope of the names.
- $(\nu \pm m_{\bar{i}})MP$: the process generates k new key pairs, m_i^+ and m_i^-, and it continues in MP, which is the scope of the key pairs.
- $MP_1 \mid MP_2$: the process denotes two meta level subprocesses running in parallel that may synchronize through communications over the network or perform actions independently.
- $!MP$: the process acts as an arbitrary number of processes MP composed in parallel.
- 0: the process is the inactive or nil process that does nothing.

The process let $X \subseteq S$ in MP is a binder of X, therefore if X is instantiated to a subset of S then every occurrence of X in the process MP is instantiated. The process $|_{i\in S}\ MP$ is a binder of i and the indexed restrictions are binders of names and key pairs.

Table 7 Syntax of meta level LySA calculus

$mx ::= x_{\bar{i}}$	
$ME ::=$	*MTerm*
$n_{\bar{i}}$	
mx	
$m_{\bar{i}}^+$	
$m_{\bar{i}}^-$	
$\{ME_1,\dots,ME_k\}_{ME_0}$	
$\{\!|\ ME_1,\dots,ME_k\ |\!\}_{ME_0}$	
$MP ::=$	*MProc*
$|_{i\in S}\ MP$	
let $X \subseteq S$ in MP	
$(\nu_{\bar{i}\in\bar{S}}\ n_{\overline{ai}})MP$	
$(\nu_{\pm\bar{i}\in\bar{S}}\ m_{\overline{ai}})MP$	
$\langle ME_1,\dots,ME_k\rangle.MP$	
$(ME_1,\dots,ME_j;mx_{j+1},\dots,mx_k).MP$	
decrypt ME as $\{ME_1,\dots,ME_j;mx_{j+1},\dots,mx_k\}_{ME_0}$ in MP	
decrypt ME as $\{\!|\ ME_1,\dots,ME_j;mx_{j+1},\dots,mx_k\ |\!\}_{ME_0}$ in MP	
$(\nu\ n_{\bar{i}})MP$	
$(\nu \pm m_{\bar{i}})MP$	
$MP_1 \mid MP_2$	
$!MP$	
0	

An instantiation relation, written $MP \rightarrow_{\mathscr{I}} P$, is introduced to describe that a process P is an instance of a meta level process MP, as depicted in Table 8.

The rule (ILet) allows the meta level to instantiate to all the object level processes P that are in some finite subset of the set S. The rule (IIPar) instantiates the process $|_{i\in S}\, MP$ to be the parallel composition of processes for each of the indexes in the set S. The rules (IINew) and (IIANew) instantiate the indexed restrictions to the restrictions of the names for all the values in the set $\{\overline{a1},\ldots,\overline{ak}\}$. The rules (IOut), (IInp), (IDec), (IADec), (INew), (IANew), (IRep), (IPar) and (INil) are instantiations of their subprocesses.

Table 8 Instantiation relation $MP \rightarrow_{\mathscr{I}} P$

(ILet)	$\dfrac{MP[X \mapsto S'] \Rrightarrow P}{\text{let } X \subseteq S \text{ in } MP \Rrightarrow P}$ if $S' \subseteq_{fin} S$
(IIPar)	$\dfrac{MP[i \mapsto a_1] \Rrightarrow P_1 \ldots MP[i \mapsto a_k] \Rrightarrow P_k}{\vert_{i\in\{a_1,\ldots,a_k\}}\, MP \Rrightarrow P_1 \mid \ldots \mid P_k}$
(IINew)	$\dfrac{MP \Rrightarrow P}{(\nu_{\bar{i}\in\{\overline{a_1},\ldots,\overline{a_k}\}}\, n_{\overline{ai}})MP \Rrightarrow (\nu\, n_{\overline{aa_1}})\ldots(\nu\, n_{\overline{aa_k}})P}$
(IIANew)	$\dfrac{MP \Rrightarrow P}{(\nu_{\pm\bar{i}\in\{\overline{a_1},\ldots,\overline{a_k}\}}\, m_{\overline{ai}})MP \Rrightarrow (\nu \pm\, m_{\overline{aa_1}})\ldots(\nu\, m_{\overline{aa_k}})P}$
(IOut)	$\dfrac{MP \Rrightarrow P}{\langle ME_1,\ldots,ME_k\rangle.MP \Rrightarrow \langle ME_1,\ldots,ME_k\rangle.P}$
(IInp)	$\dfrac{MP \Rrightarrow P}{(ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k).MP \Rrightarrow}$ $(ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k).P$
(IDec)	$\dfrac{MP \Rrightarrow P}{\text{decrypt } ME \text{ as } \{ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k\}_{ME_0} \text{ in } MP \Rrightarrow}$ decrypt ME as $\{ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k\}_{ME_0}$ in P
(IADec)	$\dfrac{MP \Rrightarrow P}{\text{decrypt } ME \text{ as } \{\mid ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k \mid\}_{ME_0} \text{ in } MP \Rrightarrow}$ decrypt ME as $\{\mid ME_1,\ldots,ME_j;mx_{j+1},\ldots,mx_k \mid\}_{ME_0}$ in P
(INew)	$\dfrac{MP \Rrightarrow P}{(\nu\, n_{\bar{a}})MP \Rrightarrow (\nu\, n_{\bar{a}})P}$
(IANew)	$\dfrac{MP \Rrightarrow P}{(\nu \pm m_{\bar{a}})MP \Rrightarrow (\nu \pm m_{\bar{a}})P}$
(IRep)	$\dfrac{MP \Rrightarrow P}{!MP \Rrightarrow !P}$
(IPar)	$\dfrac{MP_1 \Rrightarrow P_1 \quad MP_2 \Rrightarrow P_2}{MP_1 \mid MP_2 \Rrightarrow P_1 \mid P_2}$
(INil)	$0 \Rrightarrow 0$

Example 1. Let us introduce a known non-repudiation protocol, namely the Zhou-Gollmann protocol [12], which is the following:

A → B : f_{NRO}, B, L, C, NRO
B → A : f_{NRR}, A, L, NRR
A → TTP : f_{SUB}, B, L, K, sub_K
B ↔ TTP : $f_{CON}, A, B, L, K, con\_K$

A $\leftrightarrow$ TTP : $f_{CON}, A, B, L, K, con\_K$

where:

- A is the originator of the non-repudiation exchange;
- B is the recipient of the non-repudiation exchange;
- TTP is the on-line trusted third party providing network services accessible to the public;
- M is the message sent from A to B;
- C is the encryption for the message M under a key K;
- K is the message key defined by A;
- L is a unique label that links all messages of a particular protocol run together;
- $NRO = Sig_A(f_{NRO}, B, L, C)$ is the non-repudiation of origin for M;
- $NRR = Sig_B(f_{NRR}, A, L, C)$ is the non-repudiation of receipt for M;
- $sub\_K = Sig_A(f_{SUB}, B, L, K)$ is the proof of submission of K;
- $con\_K = Sig_{TTP}(f_{CON}, A, B, L, K)$ is the confirmation of K issued by TTP;
- f_* is a flag which expresses the aim of the message (the sender wants to give a proof of origin NRO / receipt NRR / submission SUB / confirmation con_K).

The encoding is the following, where three key pairs ($AK^{\pm}$ for A, $BK^{\pm}$ for B, and $TTP^{\pm}$ for the trusted third party) and a symmetric key (SK) are used:

$$
\begin{array}{l}
(\nu \pm TTP)(\nu \pm AK)(\nu \pm BK)(\\
\\
\quad !(\nu\, SK)(\nu\, L)(\nu\, M) \\
\quad \langle f_{NRO}, B, L, \{M\}_{SK}, \{|\, f_{NRO}, B, L, \{M\}_{SK} \,|\}_{AK^-} \rangle. \\
\quad (f_{NRR}, A, L; xNRR). \\
\quad \text{decrypt } xNRR \text{ as } \{|\, f_{NRR}, A, L, \{M\}_{SK}; |\}_{BK^+} \text{ in} \\
\quad \langle f_{SUB}, B, L, SK, \{|\, f_{SUB}, B, L, SK \,|\}_{AK^-} \rangle. \\
\quad (f_{CON}, A, B, L, SK; xCon). \\
\quad \text{decrypt } xCon \text{ as } \{|\, f_{CON}, A, B, L, SK; |\}_{TTP^+} \text{ in } 0 \\
\\
| \quad !(f_{NRO}, B; xL, xEnMsg, xNRO). \\
\quad \text{decrypt } xNRO \text{ as } \{|\, f_{NRO}, B, xL, xEnMsg; |\}_{AK^+} \text{ in} \\
\quad \langle f_{NRR}, A, xL, \{|\, f_{NRR}, A, xL, xEnMsg \,|\}_{BK^-} \rangle. \\
\quad (f_{CON}, A, BxL; xK, xCon). \\
\quad \text{decrypt } xCon \text{ as } \{|\, f_{CON}, A, B, xL, xK; |\}_{TTP^+} \text{ in} \\
\quad \text{decrypt } xEnMsg \text{ as } \{; xMsg\}_{xK} \text{ in } 0 \\
\\
| \quad !(f_{SUB}, B; xL, xSK, xSub). \\
\quad \text{decrypt } xSub \text{ as } \{|\, f_{SUB}, B, xL, xSK; |\}_{AK^+} \text{ in} \\
\quad \langle f_{CON}, A, B, xL, xSK, \{|\, f_{CON}, A, B, xL, xSK \,|\}_{TTP^-} \rangle. \\
\quad \langle f_{CON}, A, B, xL, xSK, \{|\, f_{CON}, A, B, xL, xSK \,|\}_{TTP^-} \rangle.0 \\
)
\end{array}
$$

where the restrictions $(\nu \pm TTP)$, $(\nu \pm AK)$, and $(\nu \pm BK)$ are used only once with scope in the whole protocol, i.e. there are three private keys known only by the owners and three public keys known by all the principals in the network.

In this scenario we have modelled only three principals, each one with a specific role, but this is not realistic. In fact, in the global network there are many principals and this gives chances to an attack. Therefore we have to extend the protocol above with multiple principals, simply indexing each name, each variable and each parallel composition construct. We consider a scenario in which there are a trusted third party (an honest principal) and many initiators and responders. The set X contains both initiators and responders, so each principal can be one or the other. The resulting protocol is the following:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{let } X \subseteq S \text{ in } (\nu_{\pm i \in X}\, AK_i)(\nu \pm\, TTP)(} \\
\quad |_{i\in X}|_{j\in X} & !(\nu\, SK_{ij})(\nu\, L_{ij})(\nu\, M_{ij}) \\
& \langle f_{NRO}, I_j, L_{ij}, \{M_{ij}\}_{SK_{ij}}, \{\mid f_{NRO}, I_j, L_{ij}, \{M_{ij}\}_{SK_{ij}} \mid\}_{AK_i^-}\rangle. \\
& (f_{NRR}, I_i, L_{ij}; xNRR_{ij}). \\
& \text{decrypt } xNRR_{ij} \text{ as } \{\mid f_{NRR}, I_i, L_{ij}, \{M_{ij}\}_{SK_{ij}}; \mid\}_{AK_j^+} \text{ in} \\
& \langle f_{SUB}, I_j, L_{ij}, SK_{ij}, \{\mid f_{SUB}, I_j, L_{ij}, SK_{ij} \mid\}_{AK_i^-}\rangle. \\
& (f_{CON}, I_i, I_j, L_{ij}, SK_{ij}; xCon_{ij}). \\
& \text{decrypt } xCon_{ij} \text{ as } \{\mid f_{CON}, I_i, I_j, L_{ij}, SK_{ij}; \mid\}_{TTP^+} \text{ in } 0 \\
\\
\quad ||_{i\in X}|_{j\in X} & !(f_{NRO}, I_j; xL_{ij}, xEnMsg_{ij}, xNRO_{ij}). \\
& \text{decrypt } xNRO_{ij} \text{ as } \{\mid f_{NRO}, I_j, xL_{ij}, xEnMsg_{ij}; \mid\}_{AK_i^+} \text{ in} \\
& \langle f_{NRR}, I_i, xL_{ij}, \{\mid f_{NRR}, I_i, xL_{ij}, xEnMsg_{ij} \mid\}_{AK_j^-}\rangle. \\
& (f_{CON}, I_i, I_j, xL_{ij}; xK_{ij}, xCon_{ij}). \\
& \text{decrypt } xCon_{ij} \text{ as } \{\mid f_{CON}, I_i, I_j, xL_{ij}, xK_{ij}; \mid\}_{TTP^+} \text{ in} \\
& \text{decrypt } xEnMsg_{ij} \text{ as } \{; xMsg_{ij}\}_{xK_{ij}} \text{ in } 0 \\
\\
\quad ||_{i\in X}|_{j\in X} & !(f_{SUB}, I_j; xL_{ij}, xSK_{ij}, xSub_{ij}). \\
& \text{decrypt } xSub_{ij} \text{ as } \{\mid f_{SUB}, I_j, xL_{ij}, xSK_{ij}; \mid\}_{AK_i^+} \text{ in} \\
& \langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \{\mid f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij} \mid\}_{TTP^-}\rangle. \\
& \langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \{\mid f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij} \mid\}_{TTP^-}\rangle.0 \\
)
\end{array}
$$

3 Control Flow Analysis

In this section we introduce our Control Flow Analysis (CFA) as an extension of [9]. The aim of the CFA is to collect information about the behavior of a process and to store them in some data structures $\mathscr{A}$, called analysis components. To be finite, static analysis is forced to compute approximations rather than exact answers. Therefore the analysis can give false positives but it has to preserve soundness.

We will use Flow Logic settings for the specification and the proofs. It is a formalism for specifying static analysis and it focuses on the relationship between an analysis estimate and the process to be analysed, formally:

$$\mathscr{A} \models P$$

which is a predicate that holds when $\mathscr{A}$ is a description of the behavior of the process P.

CFA abstracts the executions and represents only some aspects of the behavior of a process which can also be infinite. We will prove the correctness of the analysis by showing that the analysis components $\mathscr{A}$ are such that the property they represent also holds when the process evolves. Formally:

$$\mathscr{A} \models P \wedge P \rightarrow P' \Rightarrow \mathscr{A} \models P'$$

The Flow Logic specifications can be of the following formats.

Definition 1 (Verbose Format). A Verbose Flow Logic specification records information about a process globally, by rules of the form

$$\mathscr{A} \models P \text{ iff a logic formula } \mathscr{F} \text{ holds}$$

that means that the analysis components $\mathscr{A}$ are estimates of the process P if and only if the logic formula $\mathscr{F}$ holds.

Definition 2 (Succinct Format). A Succinct Flow Logic specification records information about a process locally, by rules of the form

$$\mathscr{A} \models P : A' \text{ iff a logic formula } \mathscr{F} \text{ holds}$$

where $\mathscr{A}'$ is an analysis component that holds information only about the process P and it is not known anywhere else in the analysis.

The analysis components record canonical values from the set $\lfloor Val \rfloor$ ranged over by U to represent values generated by the same restriction. The component $\kappa \in \mathscr{P}(\lfloor Val \rfloor^*)$ collects the tuples of canonical values corresponding to the values communicated in the global network while $\rho : \lfloor Var \rfloor \rightarrow \mathscr{P}(\lfloor Val \rfloor)$ records the canonical values corresponding to the values that variables may become bound. A predicate $\rho, \kappa \models P$ says that ρ and κ are valid analysis results describing the behavior of P. To analyse the expressions it is used the form $\rho \models E : \vartheta$ to describe a set of canonical values $\vartheta \in \mathscr{P}(\lfloor Val \rfloor)$ that the expression E may evaluate.

The analysis of terms and processes is described in Table 9. The rules (AN), (ANp) and (ANm) say that names may evaluate to themselves iff the canonical names are in ϑ. The rule (AVar) says that variables may evaluate to the values described by ρ for the corresponding canonical variable. The rules (AEnc) and (AAEnc) use the analysis predicate recursively to evaluate all the subexpressions in the encryption and they require ϑ to contain all the encrypted values that can be formed combining the values that subexpressions may evaluate to. The rule (AOut) says that the expressions are evaluated and it is required that all the combinations of the values found by this evaluation are recorded in κ. The rule (AInp) says that the first j expressions in the input construct are evaluated to be the sets ϑ_i for $i = 1,\dots,j$; if the pattern match with the values in κ is successful, the remaining values of the k-tuple is recorded in ρ as possible binding of the variables and the continuation process is analysed. The rule (ASDec), (AADec) and (AASig) evaluate the expression E into the set ϑ and the first j expressions in the decryption constructs are evaluated to be the sets ϑ_i for $i = 1,\dots,j$; if the pattern match with the values in κ is successful, the remaining values of the k-tuple is recorded in ρ as possible binding of the variables and the continuation process is analysed. Notice that the original syntax [3] [6] uses only the rule (AADec) to define both asymmetric decryption and signature while we introduce here two rules imposing an order in the choice of the keys to make our analysis more efficient. The rule (ANew), (AANew), (APar) and (ARep) require that the subprocesses are analysed. The rule (ANil) deals with the trivial case.

Whenever the requirements hold, the continuation process is analysed.

Table 9 Analysis of terms and processes

(AN)	$\rho \models n : \vartheta$	iff $\lfloor n \rfloor \in \vartheta$
(ANp)	$\rho \models m^+ : \vartheta$	iff $\lfloor m^+ \rfloor \in \vartheta$
(ANm)	$\rho \models m^- : \vartheta$	iff $\lfloor m^- \rfloor \in \vartheta$
(AVar)	$\rho \models x : \vartheta$	iff $\rho(\lfloor x \rfloor) \subseteq \vartheta$
(AEnc)	$\rho \models \{E_1,\dots,E_k\}_{E_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0,\dots,U_k : \bigwedge_{i=0}^{k} U_i \in \vartheta_i$ $\Rightarrow \{U_1,\dots,U_k\}_{U_0} \in \vartheta$
(AAEnc)	$\rho \models \{\!\mid E_1,\dots,E_k \mid\!\}_{E_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0,\dots,U_k : \bigwedge_{i=0}^{k} U_i \in \vartheta_i$ $\Rightarrow \{\!\mid U_1,\dots,U_k \mid\!\}_{U_0} \in \vartheta$
(AOut)	$\rho,\kappa \models \langle E_1,\dots,E_k\rangle.P$	iff $\bigwedge_{i=1}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_1,\dots,U_k : \bigwedge_{i=1}^{k} U_i \in \vartheta_i$ $\Rightarrow (\langle U_1,\dots,U_k\rangle \in \kappa \wedge \rho,\kappa \models P)$
(AInp)	$\rho,\kappa \models (E_1,\dots,E_j;x_{j+1},\dots,x_k).P$	iff $\bigwedge_{i=1}^{j} \rho \models E_i : \vartheta_i \wedge \forall \langle U_1,\dots,U_k\rangle \in \kappa : \bigwedge_{i=1}^{j} U_i \in \vartheta_i$ $\Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models P)$
(ASDec)	$\rho,\kappa \models$ decrypt E as $\{E_1,\dots,E_j;x_{j+1},\dots,x_k\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{U_1,\dots,U_k\}_{U_0} \in \vartheta$ $\wedge \bigwedge_{i=0}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models P)$
(AADec)	$\rho,\kappa \models$ decrypt E as $\{\!\mid E_1,\dots,E_j;x_{j+1},\dots,x_k \mid\!\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{\!\mid U_1,\dots,U_k \mid\!\}_{U_0} \in \vartheta :$ $\forall U_0' \in \vartheta_0 : \forall (m^+,m^-) : (U_0,U_0') = (\lfloor m^- \rfloor, \lfloor m^+ \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models P)$
(AASig)	$\rho,\kappa \models$ decrypt E as $\{\!\mid E_1,\dots,E_j;x_{j+1},\dots,x_k \mid\!\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{\!\mid U_1,\dots,U_k \mid\!\}_{U_0} \in \vartheta :$ $\forall U_0' \in \vartheta_0 : \forall (m^+,m^-) : (U_0,U_0') = (\lfloor m^+ \rfloor, \lfloor m^- \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models P)$
(ANew)	$\rho,\kappa \models (\nu\, n)P$	iff $\rho,\kappa \models P$
(AANew)	$\rho,\kappa \models (\nu \pm m)P$	iff $\rho,\kappa \models P$
(APar)	$\rho,\kappa \models P_1 \mid P_2$	iff $\rho,\kappa \models P_1 \wedge \rho,\kappa \models P_2$
(ARep)	$\rho,\kappa \models !P$	iff $\rho,\kappa \models P$
(ANil)	$\rho,\kappa \models 0$	iff $true$

The analysis is also defined for the meta level as an extension of the analysis seen so far and it takes the form

$$\rho, \kappa \models_{\Gamma} M$$

where $\Gamma : SetID \cup \mathscr{P}(Index_{fin}) \to \mathscr{P}(Index_{fin})$ is a mapping from set identifiers to finite sets of indexes. To solve the problem of infinite object level processes we use again the canonical representation of the names. The analysis is defined in Table 10, and the new rules are explained below; the rest of the rules are similar to the ones for analysing object level (the one seen so far), except that they range over indexed names and variables.

The rule (MLet) updates Γ with the mapping $X \mapsto S'$, where S' is required to be finite and it has the same canonical names as the set S. The rule (MIPar) expresses that the analysis holds for all the processes where the index i is substituted by all the elements in $\Gamma(S)$. The rules (MINew) and (MIANew) ignore the restriction operators.

3.1 The attacker

The attacker is unique and runs its protocol $P_\bullet$ following the Dolev-Yao formula $\mathscr{F}_{RM}^{DY}$ [5]. We write $P_{sys} \mid P_\bullet$ to show that an arbitrary attacker controls the whole network while principals exchange messages using the protocol. A protocol process P_{sys} has type whenever it is close, all its free names are in $\mathscr{N}_f$, all the arities of the sent or received messages are in $\mathscr{A}_\kappa$ and all the arities of the encrypted or decrypted messages are in $\mathscr{A}_{Enc}$. These three sets are finite, like $\mathscr{N}_c$ and $\mathscr{X}_c$, used to collect all the names and all the variables respectively in the process P_{sys}. The attacker uses a new name, $n_\bullet \notin \mathscr{N}_c$, and a new variable, $z_\bullet \notin \mathscr{X}_c$, which do not overlap the names and the variables used by the legitimate principals. It is again considered a process with finitely many canonical names and variables. A formula $\mathscr{F}_{RM}^{DY}$ of the type $(\mathscr{N}_f, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$, which is capable of characterizing the potential effect of all the attackers $P_\bullet$ of the type $(\mathscr{N}_f, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$, is defined as the conjunction of the components in Table 11.

4 Non-Repudiation Analysis

Non-repudiation guarantees that the principals exchanging messages cannot falsely deny having sent or received the messages. This is done using evidences [7] that allow to decide unquestionably in favor of the fair principal whenever there is a dispute. In particular, non-repudiation of origin provides the recipient with proof of origin while non-repudiation of receipt provides the originator with proof of receipt. Evidences [13] should have verifiable origin, integrity and validity.

Table 10 The meta level analysis

(MLet)	$\rho,\kappa \models_{\Gamma}$ let $X \subseteq S$ in M	iff $\rho,\kappa \models_{\Gamma[X\mapsto S']} M$ where $S' \subseteq_{fin} \Gamma(S)$ and $\lfloor S' \rfloor = \lfloor \Gamma(S) \rfloor$
(MIPar)	$\rho,\kappa \models_{\Gamma} \|_{i\in S} M$	iff $\bigwedge_{a\in\Gamma(S)} \rho,\kappa \models_{\Gamma} M[i\mapsto a]$
(MINew)	$\rho,\kappa \models_{\Gamma} (\nu_{\bar{i}\in\bar{S}} n_{\overline{ai}})M$	iff $\rho,\kappa \models_{\Gamma} M$
(MIANew)	$\rho,\kappa \models_{\Gamma} (\nu_{\pm\bar{i}\in\bar{S}} m_{\overline{ai}})M$	iff $\rho,\kappa \models_{\Gamma} M$
(MN)	$\rho \models n_{\bar{i}} : \vartheta$	iff $\lfloor n_{\bar{i}} \rfloor \in \vartheta$
(MNp)	$\rho \models m^{+}_{\bar{i}} : \vartheta$	iff $\lfloor m^{+}_{\bar{i}} \rfloor \in \vartheta$
(MNm)	$\rho \models m^{-}_{\bar{i}} : \vartheta$	iff $\lfloor m^{-}_{\bar{i}} \rfloor \in \vartheta$
(MVar)	$\rho \models x_{\bar{i}} : \vartheta$	iff $\rho(\lfloor x_{\bar{i}} \rfloor) \subseteq \vartheta$
(MEnc)	$\rho \models \{ME_1,\ldots,ME_k\}_{ME_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models ME_i : \vartheta_i \wedge \forall U_0,\ldots,U_k :$ $\bigwedge_{i=0}^{k} U_i \in \vartheta_i \Rightarrow \{U_1,\ldots,U_k\}_{U_0} \in \vartheta$
(AAEnc)	$\rho \models \{\!\mid ME_1,\ldots,ME_k \mid\!\}_{ME_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models ME_i : \vartheta_i \wedge \forall U_0,\ldots,U_k :$ $\bigwedge_{i=0}^{k} U_i \in \vartheta_i \Rightarrow \{\!\mid U_1,\ldots,U_k \mid\!\}_{U_0} \in \vartheta$
(MOut)	$\rho,\kappa \models_{\Gamma} \langle ME_1,\ldots,ME_k\rangle.M$	iff $\bigwedge_{i=1}^{k} \rho \models ME_i : \vartheta_i \wedge \forall\, U_1,\ldots,U_k :$ $\bigwedge_{i=1}^{k} U_i \in \vartheta_i \Rightarrow \langle U_1,\ldots,U_k\rangle \in \kappa \wedge \rho,\kappa \models_{\Gamma} M$
(MInp)	$\rho,\kappa \models_{\Gamma} (ME_1,\ldots,ME_j; x_{j+1},\ldots,x_k).M$	iff $\bigwedge_{i=1}^{j} \rho \models ME_i : \vartheta_i \wedge \forall \langle U_1,\ldots,U_k\rangle \in \kappa : \bigwedge_{i=1}^{j} U_i \in \vartheta_i$ $\Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models_{\Gamma} M)$
(ASDec)	$\rho,\kappa \models_{\Gamma}$ decrypt ME as $\{ME_1,\ldots,ME_j; x_{j+1},\ldots,x_k\}_{ME_0}$ in M	iff $\rho \models ME : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models ME_i : \vartheta_i \wedge$ $\forall \{U_1,\ldots,U_k\}_{U_0} \in \vartheta \wedge \bigwedge_{i=0}^{j} U_i \in \vartheta_i$ $\Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models_{\Gamma} M)$
(AADec)	$\rho,\kappa \models_{\Gamma}$ decrypt ME as $\{\!\mid ME_1,\ldots,ME_j; x_{j+1},\ldots,x_k \mid\!\}_{ME_0}$ in M	iff $\rho \models_{\Gamma} ME : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models_{\Gamma} ME_i : \vartheta_i \wedge$ $\forall \{\!\mid U_1,\ldots,U_k \mid\!\}_{U_0} \in \vartheta : \forall U'_0 \in \vartheta_0 :$ $\forall (m^+,m^-) : (U_0,U'_0) = (\lfloor m^- \rfloor, \lfloor m^- \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models_{\Gamma} M)$
(AASig)	$\rho,\kappa \models_{\Gamma}$ decrypt ME as $\{\!\mid ME_1,\ldots,ME_j; x_{j+1},\ldots,x_k \mid\!\}_{ME_0}$ in M	iff $\rho \models_{\Gamma} ME : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models ME_i : \vartheta_i \wedge$ $\forall \{\!\mid U_1,\ldots,U_k \mid\!\}_{U_0} \in \vartheta : \forall U'_0 \in \vartheta_0 :$ $\forall (m^+,m^-) : (U_0,U'_0) = (\lfloor m^+ \rfloor, \lfloor m^- \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa \models_{\Gamma} M)$
(ANew)	$\rho,\kappa \models_{\Gamma} (\nu\, n_i)M$	iff $\rho,\kappa \models_{\Gamma} M$
(AANew)	$\rho,\kappa \models_{\Gamma} (\nu \pm m_i)M$	iff $\rho,\kappa \models_{\Gamma} M$
(APar)	$\rho,\kappa \models_{\Gamma} M_1 \mid M_2$	iff $\rho,\kappa \models_{\Gamma} M_1 \wedge \rho,\kappa \models M_2$
(ARep)	$\rho,\kappa \models_{\Gamma} !M$	iff $\rho,\kappa \models_{\Gamma} M$
(ANil)	$\rho,\kappa \models_{\Gamma} 0$	iff *true*

The syntax of the process calculus LySa has to be extended to guarantee, given a protocol, the non-repudiation property, i.e. authentication (only the sender of the message can create it), integrity and freshness. This is done using electronic signatures and unique identifiers for users and sessions. To this aim, we introduce two sets, used in the body of the messages to collect information that will be useful to perform the analysis: ID, where $id \in ID$ is a unique identifier for a principal, and NR, where $nr \in NR$ says that non-repudiation property is required for that part of the message nr. To include this sets in our analysis, a redefinition of the syntax of LySa

Table 11 The attacker's capabilities

(1) The attacker may learn by eavesdropping

$$\bigwedge_{k\in\mathscr{A}_\kappa} \forall\langle V_1,\ldots,V_k\rangle \in \kappa : \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet)$$

(2) The attacker may learn by decrypting messages with keys already known

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{V_1,\ldots,V_k\}_{V_0} \in \rho(z_\bullet) : V_0 \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{\!|\, V_1,\ldots,V_k \,|\!\}_{m^+} \in \rho(z_\bullet) : m^- \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{\!|\, V_1,\ldots,V_k \,|\!\}_{m^-} \in \rho(z_\bullet) : m^+ \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet)$$

(3) The attacker may construct new encryptions using the keys known

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall V_0,\ldots,V_k : \bigwedge_{i=0}^{k} V_i \in \rho(z_\bullet) \Rightarrow \{V_1,\ldots,V_k\}_{V_0} \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall m^+, V_1,\ldots,V_k : m^+ \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet) \Rightarrow \{\!|\, V_1,\ldots,V_k \,|\!\}_{m^+} \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall m^-, V_1,\ldots,V_k : m^- \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet) \Rightarrow \{\!|\, V_1,\ldots,V_k \,|\!\}_{m^-} \in \rho(z_\bullet)$$

(4) The attacker may actively forge new communications

$$\bigwedge_{k\in\mathscr{A}_\kappa} \forall V_1,\ldots,V_k : \bigwedge_{i=1}^{k} V_i \in \rho(z_\bullet) \Rightarrow \langle V_1,\ldots,V_k\rangle \in \kappa$$

(5) The attacker initially has some knowledge
$\{n_\bullet, m_\bullet^\pm\} \cup \mathscr{N}_f \subseteq \rho(z_\bullet)$

is required, as shown in Table 12. Observe that, with respect to the LySA calculus in 1, a unique identifier u is associated to encryption and decryption and an $id \in ID$ is associated to public and private keys to specify the principal that encrypts a given message. The redefinition is obtained applying the function $\mathscr{G}$ to the processes of the protocol analysed, that acts recursively on the subprocesses and redefines subterms using another function, called $\mathscr{F}$. The definition of the functions $\mathscr{F}$ and $\mathscr{G}$, that map standard terms and processes into the extended ones, is shown in Table 13. Notice that the functions provide a new syntax in which:

- *id*s are attached whenever an asymmetric key appears;
- a session identifier u is attached to each encryption and decryption;
- parallel composition assigns a different *id* to each process, because the two processes belong to a different user;
- replication has a particular form that the semantic rules use to create replications of the process with different *id*s (that has to be unique).

Notice that we have generalized the approach [6] proposed by H. Gao to provide freshness property in a protocol. Indeed, the author defines two functions to attach a session identifier to each statement; then, he redefines the semantics, using the

Table 12 Syntax of LySA calculus extended with principal identifiers

$\varepsilon ::=$	*terms*
n	name
x	variable
$[m^+]_{id}$	public key
$[m^-]_{id}$	private key
$\{\varepsilon_1, \ldots, \varepsilon_k\}_{\varepsilon_0}$	symmetric encryption
$\{\mid \varepsilon_1, \ldots, \varepsilon_k \mid\}^u_{\varepsilon_0}$	asymmetric encryption
$\mathscr{P} ::=$	*processes*
$\langle \varepsilon_1, \ldots, \varepsilon_k \rangle.\mathscr{P}$	output
$(\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k).\mathscr{P}$	input
decrypt ε as $\{\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\}_{\varepsilon_0}$ in $\mathscr{P}$	symmetric decryption
decrypt ε as $\{\mid \varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k \mid\}^u_{\varepsilon_0}$ in $\mathscr{P}$	asymmetric decryption
$(\nu\, n)\mathscr{P}$	restriction
$(\nu \pm [m]_{id})\mathscr{P}$	pair restriction
$\mathscr{P}_1 \mid \mathscr{P}_2$	parallel composition
$[!P]_{id}$	replication
0	nil

functions to avoid to redefine the structural congruence. In our analysis, because of the redefinition of the latter, we have not to modify significantly the reduction semantics, except that the rule (NRNRep) takes advantage of a particular syntax that allows to attach different and unique identifiers to each process. has to be removed

Table 13 Functions $\mathscr{F}$ and $\mathscr{G}$

$\mathscr{F} : E \times ID \to \varepsilon$
- $\mathscr{F}(n, id) = n$
- $\mathscr{F}(x, id) = x$
- $\mathscr{F}(m^+, id) = [m^+]_{id}$
- $\mathscr{F}(m^-, id) = [m^-]_{id}$
- $\mathscr{F}(\{E_1, \ldots, E_k\}_{E_0}, id) = \{\mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_k, id)\}_{\mathscr{F}(E_0, id)}$
- $\mathscr{F}(\{\mid E_1, \ldots, E_k \mid\}_{E_0}, id) = \{\mid \mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_k, id) \mid\}^u_{\mathscr{F}(E_0, id)}$

$\mathscr{G} : P \times ID \to \mathscr{P}$
- $\mathscr{G}(\langle E_1, \ldots, E_k \rangle.\mathscr{P}, id) = \langle \mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_k, id) \rangle.\mathscr{G}(P, id)$
- $\mathscr{G}((E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P, id) =$
  $(\mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_j, id); x_{j+1}, \ldots, x_k).\mathscr{G}(P, id)$
- $\mathscr{G}($decrypt E as $\{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}_{E_0}$ in $P, id) =$
  decrypt $\mathscr{F}(E, id)$ as $\{\mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_j, id); x_{j+1}, \ldots, x_k\}_{\mathscr{F}(E_0, id)}$ in $\mathscr{G}(P, id)$
- $\mathscr{G}($decrypt E as $\{\mid E_1, \ldots, E_j; x_{j+1}, \ldots, x_k \mid\}^u_{E_0}$ in $P, id) =$
  decrypt $\mathscr{F}(E, id)$ as $\{\mid \mathscr{F}(E_1, id), \ldots, \mathscr{F}(E_j, id); x_{j+1}, \ldots, x_k \mid\}^u_{\mathscr{F}(E_0, id)}$ in $\mathscr{G}(P, id)$
- $\mathscr{G}((\nu\, n)P, id) = (\nu\, n)\mathscr{G}(P, id)$
- $\mathscr{G}((\nu \pm m)P, id) = (\nu \pm [m]_{id})\mathscr{G}(P, id)$
- $\mathscr{G}(P \mid Q, id) = \mathscr{G}(P, id) \mid \mathscr{G}(Q, id')$
- $\mathscr{G}(!P, id) = [!P]_{id}$
- $\mathscr{G}(0, id) = 0$

because the structural equivalence does not hold in this case. The rule (NRNRep) will appropriately treat the behavior of the replication statement, as reported in Table 17. Finally, we have to add the following annotations to the signatures:

- [from id] is associated to encryption and it means that the recipient expects a message from id.
- [check NR] is associated to decryption and it means that for all the elements of the set NR, non-repudiation property must be guaranteed. It is interesting to notice that the elements in the set NR can specify a part of the message, not necessarily the whole message, according to the definition of non-repudiation.

The syntax of asymmetric encryption and decryption becomes:

- $\{|\ \varepsilon_1,\ldots,\varepsilon_k\ |\}^u_{\varepsilon_0}[\text{from } id]$
- decrypt ε as $\{|\ \varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\ |\}^u_{\varepsilon_0}$ [check NR] in $\mathscr{P}$

Notice that the annotation [from id] and the label u have a different role in the analysis. The first says that the principal who encrypted the message must be the same specified in the label associated to the private key used, while the second expresses that the message has to belong to a precise session.

In practice, when there is a violation due to the ids, it means that the attacker encrypted a message and sent it to a principal who expected it from another principal (remember that the attacker can even use a key known different from his key). Instead, when there is a violation due to the labels u, it means that the attacker made a replay attack using a message exchanged in a previous session.

4.1 Dynamic Property

To guarantee the dynamic property, the values have to be redefined into $NRVal$, attaching the identifiers to the asymmetric key pairs and the annotations in the encryption constructs as shown below:

$$
\begin{array}{rl}
NRV ::= & n \\
| & [m^+]_{id} \\
| & [m^-]_{id} \\
| & \{NRV_1,\ldots,NRV_k\}_{NRV_0} \\
| & \{|\ NRV_1,\ldots,NRV_k\ |\}^u_{NRV_0}[\text{from } id]
\end{array}
$$

Furthermore, our extension involves redefinition of the semantics, of free names, of structural congruence, and of α-equivalence, as described in the Tables 17, 14, 15, 16, respectively.

Notice that there are the following differences between the previous semantics and the one used in the analysis:

- The asymmetric encryption and decryption are redefined adding a session identifier u, an identifier that shows who has encrypted a given cipher message, and the annotations above.

- New terms ε and processes $\mathscr{P}$ are used instead of the previous, T and P, which do not carry annotations.
- The process $!\mathscr{P}$ is not structurally equivalent to $\mathscr{P} \mid !\mathscr{P}$, because of the recursive definition of the function $\mathscr{G}$.
- The rule (NRNRep) assures that a different id is associated at each process, therefore the same principal will have the same id for each encryption; id' has to be unique.

We use the reference monitor semantics ($\rightarrow_{RM}$), an extension of the standard semantics ($\rightarrow_{\mathscr{R}}$), to check the non-repudiation property. Taking advantage of annotations, it forces some requirements and, if they are not meet, the process execution is aborted.

The reference monitor semantics $P \rightarrow_{RM} P'$ takes annotations into account and defines RM as

$$RM(id, id', u, u', \{NRV_1, \ldots, NRV_n\}, NR) =$$
$$(id = id' \wedge u = u' \wedge \forall nr \in NR : nr \in \{NRV_1, \ldots, NRV_n\})$$

where $\{NRV_1, \ldots, NRV_n\}$ is a set of redefined values for non-repudiation analysis. When the reference monitor is turned on, the reduction relation $\rightarrow_{\mathscr{R}}$ checks if the requirements are met; otherwise $\mathscr{R}$ is considered $true$, i.e. the execution cannot be aborted for the requirements above, it verify only the assumptions of the standard rules.

Intuitively, we verify if the message received is encrypted by the correct sender and if it is a fresh message.

The main difference between the standard semantics and the redefined semantics is expressed by the rule used to verify a signature. In fact, when the reference monitor is turned on, the rules (NRNSig) ensures that the non-repudiation property holds for the elements specified by the annotations.

Definition 3 (Dynamic Non-Repudiation). A process $\mathscr{P}$ ensures *dynamic non-repudiation* property if for all the executions

$$\mathscr{P} \rightarrow^* \mathscr{P}' \rightarrow_{RM} \mathscr{P}''$$

$id = id'$ and $u = u'$ and $\forall nr \in NR : nr \in \{NRV_1, \ldots, NRV_k\}$ when $\mathscr{P}' \rightarrow_{RM} \mathscr{P}''$ is derived using (ASig) on

decrypt $\{\mid NRV_1, \ldots, NRV_k \mid\}^{u}_{[m^-]_{id}}$[from id'] as

$\{\mid NRV'_1, \ldots, NRV'_j; x_{j+1}, \ldots, x_k \mid\}^{u'}_{[m^+]_{id}}$[check NR] in $\mathscr{P}$

Definition 1 says that an extended process $\mathscr{P}$ ensures non-repudiation property if there is no violation in any of its execution.

4.2 Static Property

A component $\psi \subseteq \mathscr{P}(NR)$ will collect all the labels nr such that the non-repudiation property for the element nr is possibly violated.

Table 14 Redefinition of the function $fn(P)$

$fn(n)$	$\stackrel{def}{=} \{n\}$
$fn([m^+]_{id})$	$\stackrel{def}{=} \{[m^+]_{id}\}$
$fn([m^-]_{id})$	$\stackrel{def}{=} \{[m^-]_{id}\}$
$fn(x)$	$\stackrel{def}{=} \emptyset$
$fn(\{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0})$	$\stackrel{def}{=} fn(\varepsilon_0)\cup\ldots\cup fn(\varepsilon_k)$
$fn(\{\mid \varepsilon_1,\ldots,\varepsilon_k \mid\}^{u}_{\varepsilon_0}[\text{from } id])$	$\stackrel{def}{=} fn(\varepsilon_0)\cup\ldots\cup fn(\varepsilon_k)$
$fn(\langle\varepsilon_1,\ldots,\varepsilon_k\rangle.\mathscr{P})$	$\stackrel{def}{=} fn(\varepsilon_1)\cup\ldots\cup fn(\varepsilon_k)\cup fn(\mathscr{P})$
$fn((\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k).\mathscr{P})$	$\stackrel{def}{=} fn(\varepsilon_1)\cup\ldots\cup fn(\varepsilon_j)\cup fn(\mathscr{P})$
$fn(\text{decrypt } \varepsilon \text{ as } \{\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\}_{\varepsilon_0} \text{ in } \mathscr{P})$	
	$\stackrel{def}{=} fn(\varepsilon)\cup fn(\varepsilon_0)\cup\ldots\cup fn(\varepsilon_j)\cup fn(\mathscr{P})$
$fn(\text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k \mid\}^{u}_{\varepsilon_0} \text{ [check } NR] \text{ in } \mathscr{P})$	
	$\stackrel{def}{=} fn(\varepsilon)\cup fn(\varepsilon_0)\cup\ldots\cup fn(\varepsilon_j)\cup fn(\mathscr{P})$
$fn((\nu\, n)\mathscr{P})$	$\stackrel{def}{=} fn(\mathscr{P})\setminus\{n\}$
$fn((\nu \pm [m]_{id})\mathscr{P})$	$\stackrel{def}{=} fn(\mathscr{P})\setminus\{[m^+]_{id},[m^-]_{id}\}$
$fn(\mathscr{P}_1 \mid \mathscr{P}_2)$	$\stackrel{def}{=} fn(\mathscr{P}_1)\cup fn(\mathscr{P}_2)$
$fn([!P]_{id})$	$\stackrel{def}{=} fn(\mathscr{G}(P,id))$
$fn(0)$	$\stackrel{def}{=} \emptyset$

Table 15 Redefinition of the structural congruence $P \equiv P'$

$\mathscr{P} \equiv \mathscr{P}$

$\mathscr{P}_1 \equiv \mathscr{P}_2 \Rightarrow \mathscr{P}_2 \equiv \mathscr{P}_1$

$\mathscr{P}_1 \equiv \mathscr{P}_2 \wedge \mathscr{P}_2 \equiv \mathscr{P}_3 \Rightarrow \mathscr{P}_1 \equiv \mathscr{P}_3$

$$\mathscr{P}_1 \equiv \mathscr{P}_2 \Rightarrow \begin{cases} \langle\varepsilon_1,\ldots,\varepsilon_k\rangle.\mathscr{P}_1 \equiv \langle\varepsilon_1,\ldots,\varepsilon_k\rangle.\mathscr{P}_2 \\ (\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k).\mathscr{P}_1 \equiv (\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k).\mathscr{P}_2 \\ \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\}_{\varepsilon_0} \text{ in } \mathscr{P}_1 \equiv \\ \quad \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\}_{\varepsilon_0} \text{ in } \mathscr{P}_2 \\ \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k \mid\}^{u}_{\varepsilon_0} \text{ [check } NR] \text{ in } \mathscr{P}_1 \equiv \\ \quad \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k \mid\}^{u}_{\varepsilon_0} \text{ [check } NR] \text{ in } \mathscr{P}_2 \\ (\nu\, n)\mathscr{P}_1 \equiv (\nu\, n)\mathscr{P}_2 \\ (\nu \pm [m]_{id})\mathscr{P}_1 \equiv (\nu \pm [m]_{id})\mathscr{P}_2 \\ \mathscr{P}_1 \mid \mathscr{P}_3 \equiv \mathscr{P}_2 \mid \mathscr{P}_3 \end{cases}$$

$P_1 \equiv P_2 \Rightarrow [!P_1]_{id} \equiv [!P_2]_{id}$ if both P_1 and P_2 are annotated with the same id

$\mathscr{P}_1 \mid \mathscr{P}_2 \equiv \mathscr{P}_2 \mid \mathscr{P}_1$

$(\mathscr{P}_1 \mid \mathscr{P}_2) \mid \mathscr{P}_3 \equiv \mathscr{P}_1 \mid (\mathscr{P}_2 \mid \mathscr{P}_3)$

$\mathscr{P} \mid 0 \equiv \mathscr{P}$

$(\nu\, n)0 \equiv 0$

$(\nu\, n_1)(\nu\, n_2)\mathscr{P} \equiv (\nu\, n_2)(\nu\, n_1)\mathscr{P}$

$(\nu\, n)(\mathscr{P}_1 \mid \mathscr{P}_2) \equiv \mathscr{P}_1 \mid (\nu\, n)\mathscr{P}_2$ if $n \notin fn(\mathscr{P}_1)$

$(\nu \pm [m]_{id})0 \equiv 0$

$(\nu \pm [m_1]_{id})(\nu \pm [m_2]_{id})\mathscr{P} \equiv (\nu \pm [m_2]_{id})(\nu \pm [m_1]_{id})\mathscr{P}$

$(\nu \pm [m]_{id})(\mathscr{P}_1 \mid \mathscr{P}_2) \equiv \mathscr{P}_1 \mid (\nu \pm [m]_{id})\mathscr{P}_2$ if $[m^+]_{id},[m^-]_{id} \notin fn(\mathscr{P}_1)$

$(\nu \pm [m]_{id})(\nu\, n)\mathscr{P} \equiv (\nu\, n)(\nu \pm [m]_{id})\mathscr{P}$

$\mathscr{P}_1 \stackrel{\alpha}{\equiv} \mathscr{P}_2 \Rightarrow \mathscr{P}_1 \equiv \mathscr{P}_2$

Table 16 Redefinition of the α-equivalence

$\mathscr{P} \stackrel{\alpha}{\equiv} \mathscr{P}$
$\mathscr{P}_1 \stackrel{\alpha}{\equiv} \mathscr{P}_2$ implies $\mathscr{P}_2 \stackrel{\alpha}{\equiv} \mathscr{P}_1$
$\mathscr{P}_1 \stackrel{\alpha}{\equiv} \mathscr{P}_2$ and $\mathscr{P}_2 \stackrel{\alpha}{\equiv} \mathscr{P}_3$ implies $\mathscr{P}_1 \stackrel{\alpha}{\equiv} \mathscr{P}_3$
$(\nu\, n_1)\mathscr{P} \stackrel{\alpha}{\equiv} (\nu\, n_2)(\mathscr{P}[n_1 \mapsto n_2])$ if $n_2 \notin fn(\mathscr{P})$
$(\nu \pm [m_1]_{id})\mathscr{P} \stackrel{\alpha}{\equiv} (\nu \pm [m_2]_{id})(\mathscr{P}[[m_1]^+_{id} \mapsto [m_2]^+_{id}, [m_1]^-_{id} \mapsto [m_2]^-_{id}])$ if $[m_2]^+_{id}, [m_2]^-_{id} \notin fn(\mathscr{P})$

Table 17 Redefinition of the semantics of LYSA calculus

(NRNCom)	$\dfrac{\bigwedge_{i=1}^{j} NRV_i = NRV'_i}{\langle NRV_1,\ldots,NRV_k\rangle.\mathscr{P} \mid (NRV'_1,\ldots,NRV'_j;x_{j+1},\ldots,x_k).\mathscr{P}' \to_{\mathscr{R}} \mathscr{P} \mid \mathscr{P}'[NRV_{j+1}/x_{j+1},\ldots,NRV_k/x_k]}$
(NRNDec)	$\dfrac{\bigwedge_{i=0}^{j} NRV_i = NRV'_i}{\text{decrypt } \{NRV_1,\ldots,NRV_k\}_{NRV_0} \text{ as } \{NRV'_1,\ldots,NRV'_j;x_{j+1},\ldots,x_k\}_{NRV'_0}[\text{check } NR] \text{ in } \mathscr{P} \to_{\mathscr{R}} \mathscr{P}[NRV_{j+1}/x_{j+1},\ldots,NRV_k/x_k]}$
(NRNADec)	$\dfrac{\bigwedge_{i=1}^{j} NRV_i = NRV'_i}{\text{decrypt } \{\!\mid NRV_1,\ldots,NRV_k \mid\!\}^{u}_{[m^+]_{id}}[\text{from } id'] \text{ as } \{\!\mid NRV'_1,\ldots,NRV'_j;x_{j+1},\ldots,x_k \mid\!\}^{u'}_{[m^-]_{id}}[\text{check } NR] \text{ in } \mathscr{P} \to_{\mathscr{R}} \mathscr{P}[NRV_{j+1}/x_{j+1},\ldots,NRV_k/x_k]}$
(NRNSig)	$\dfrac{\bigwedge_{i=1}^{j} NRV_i = NRV'_i \wedge RM(id,id',u,u',\{NRV_{j+1},\ldots,NRV_k\},NR)}{\text{decrypt } \{\!\mid NRV_1,\ldots,NRV_k \mid\!\}^{u}_{[m^-]_{id}}[\text{from } id'] \text{ as } \{\!\mid NRV'_1,\ldots,NRV'_j;x_{j+1},\ldots,x_k \mid\!\}^{u'}_{[m^+]_{id}}[\text{check } NR] \text{ in } \mathscr{P} \to_{\mathscr{R}} \mathscr{P}[NRV_{j+1}/x_{j+1},\ldots,NRV_k/x_k]}$
(NRNNew)	$\dfrac{\mathscr{P} \to_{\mathscr{R}} \mathscr{P}'}{(\nu\, n)\mathscr{P} \to_{\mathscr{R}} (\nu\, n)\mathscr{P}'}$
(NRNANew)	$\dfrac{\mathscr{P} \to_{\mathscr{R}} \mathscr{P}'}{(\nu \pm [m]_{id})\mathscr{P} \to_{\mathscr{R}} (\nu \pm [m]_{id})\mathscr{P}'}$
(NRNPar)	$\dfrac{\mathscr{P}_1 \to_{\mathscr{R}} \mathscr{P}'_1}{\mathscr{P}_1 \mid \mathscr{P}_2 \to_{\mathscr{R}} \mathscr{P}'_1 \mid \mathscr{P}_2}$
(NRNCongr)	$\dfrac{\mathscr{P} \equiv \mathscr{P}' \wedge \mathscr{P}' \to_{\mathscr{R}} \mathscr{P}'' \wedge \mathscr{P}'' \equiv \mathscr{P}'''}{\mathscr{P} \to_{\mathscr{R}} \mathscr{P}'''}$
(NRNRep)	$[!P]_{id} \to_{\mathscr{R}} \mathscr{G}(P,id) \mid [!P]_{id'}$

The $\propto$ operator is introduced to ignore the extension of the syntax and is defined as:

$NRV \propto \vartheta$ iff there exists $V \in Val$ such that $NRV = V$ and $V \in \vartheta$
where the relation $NRV = V$ is defined to be the least equivalence between an element in *NRVal* and an element in *Val* that inductively ignores the identifiers and the annotations.

The analyses of the terms and of the processes are shown in the Tables 18 and 19. The rule (NRASig) checks the non-repudiation property whenever a signature is verified.

Table 18 Non-repudiation analysis of terms $\rho \vdash \varepsilon : \vartheta$

(NRAN)	$\rho \models n : \vartheta$	iff $\lfloor n \rfloor \in \vartheta$
(NRANp)	$\rho \models [m^+]_{id} : \vartheta$	iff $[\lfloor m^+ \rfloor]_{id} \propto \vartheta$
(NRANm)	$\rho \models [m^-]_{id} : \vartheta$	iff $[\lfloor m^- \rfloor]_{id} \propto \vartheta$
(NRAVar)	$\rho \models x : \vartheta$	iff $\rho(\lfloor x \rfloor) \subseteq \vartheta$
(NRAEnc)	$\rho \models \{\varepsilon_1,\dots,\varepsilon_k\}_{\varepsilon_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall NRV_0,\dots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow$ $\{NRV_1,\dots,NRV_k\}_{NRV_0} \propto \vartheta$
(NRAAEnc)	$\rho \models \{\mid \varepsilon_1,\dots,\varepsilon_k \mid\}^{u}_{\varepsilon_0}[\text{from } id] : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall NRV_0,\dots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow$ $\{\mid NRV_1,\dots,NRV_k \mid\}^{u}_{[NRV_0]_{id'}}[\text{from } \lfloor id \rfloor] \propto \vartheta$

Table 19 Non-repudiation analysis of processes $\rho, \kappa, \psi \models \mathscr{P}$

(NRAOut)	$\rho,\kappa,\psi \models \langle \varepsilon_1,\dots,\varepsilon_k \rangle.\mathscr{P}$ iff $\bigwedge_{i=1}^{k} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall NRV_1,\dots,NRV_k : \bigwedge_{i=1}^{k} NRV_i \propto \vartheta_i \Rightarrow$ $(\langle NRV_1,\dots,NRV_k \rangle \in \kappa \wedge \rho,\kappa,\psi \models \mathscr{P})$
(NRAInp)	$\rho,\kappa,\psi \models (\varepsilon_1,\dots,\varepsilon_j; x_{j+1},\dots,x_k).\mathscr{P}$ iff $\bigwedge_{i=1}^{j} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall \langle NRV_1,\dots,NRV_k \rangle \in \kappa : \bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i \Rightarrow$ $(\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa,\psi \models \mathscr{P})$
(NRADec)	$\rho,\kappa,\psi \models \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1,\dots,\varepsilon_j; x_{j+1},\dots,x_k\}_{\varepsilon_0} \text{ in } \mathscr{P}$ iff $\rho \models \varepsilon : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall \{NRV_1,\dots,NRV_k\}_{NRV_0} \propto \vartheta \wedge \bigwedge_{i=0}^{j} NRV_i \propto \vartheta_i \Rightarrow$ $(\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa,\psi \models \mathscr{P})$
(NRAADec)	$\rho,\kappa,\psi \models \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1,\dots,\varepsilon_j; x_{j+1},\dots,x_k \mid\}^{u'}_{\varepsilon_0}[\text{check } NR] \text{ in } \mathscr{P}$ iff $\rho \models \varepsilon : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall \{\mid NRV_1,\dots,NRV_k \mid\}^{u}_{NRV_0}[\text{from } \lfloor id \rfloor] \propto \vartheta :$ $\forall NRV'_0 \propto \vartheta_0 : \forall (m^+, m^-) : (NRV_0, NRV'_0) =$ $([\lfloor m^- \rfloor]_{id'}, [\lfloor m^+ \rfloor]_{id'}) \wedge \bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i \Rightarrow$ $(\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa,\psi \models \mathscr{P})$
(NRASig)	$\rho,\kappa,\psi \models \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1,\dots,\varepsilon_j; x_{j+1},\dots,x_k \mid\}^{u'}_{\varepsilon_0}[\text{check } NR] \text{ in } \mathscr{P}$ iff $\rho \models \varepsilon : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \varepsilon_i : \vartheta_i \wedge$ $\forall \{\mid NRV_1,\dots,NRV_k \mid\}^{u}_{NRV_0}[\text{from } \lfloor id \rfloor] \propto \vartheta :$ $\forall NRV'_0 \propto \vartheta_0 : \forall m^+, m^-, id, id' : (NRV_0, NRV'_0) =$ $([\lfloor m^+ \rfloor]_{id'}, [\lfloor m^- \rfloor]_{id'}) \wedge \bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i \Rightarrow$ $(\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa,\psi \models \mathscr{P} \wedge$ $\forall nr \in NR : (id \neq id' \vee u \neq u' \vee$ $nr \notin \{NRV_{j+1},\dots,NRV_k\}) \Rightarrow \lfloor nr \rfloor \in \psi)$
(NRANew)	$\rho,\kappa,\psi \models (\nu\, n)\mathscr{P}$ iff $\rho,\kappa,\psi \models \mathscr{P}$
(NRAANew)	$\rho,\kappa,\psi \models (\nu \pm m)\mathscr{P}$ iff $\rho,\kappa,\psi \models \mathscr{P}$
(NRAPar)	$\rho,\kappa,\psi \models \mathscr{P}_1 \mid \mathscr{P}_2$ iff $\rho,\kappa,\psi \models \mathscr{P}_1 \wedge \rho,\kappa,\psi \models \mathscr{P}_2$
(NRARep)	$\rho,\kappa,\psi \models [!P]_{id}$ iff $\rho,\kappa,\psi \models \mathscr{G}(P, id)$
(NRANil)	$\rho,\kappa,\psi \models 0$ iff $true$

To prove the correctness of our analysis we must prove that it respects the extended operational semantics of LYSA, i.e. if $\rho, \kappa, \psi \models \mathscr{P}$ then the triple (ρ, κ, ψ) is a valid estimate for all the states passed through in a computation of $\mathscr{P}$. Furthermore, we prove that when ψ is empty, then the reference monitor is useless.

Our proof uses three lemmas, defined and proved below. The first and the second show that estimates are resistant to substitution of closed terms for variables, both in the terms and in the processes; the third says that an estimate for an extended process $\mathscr{P}$ is valid for every process congruent to $\mathscr{P}$.

Lemma 1 (Substitution in expression). *If* $\rho \models \varepsilon : \vartheta$ *and* $\varepsilon' \in \rho(x)$ *then* $\rho \models \varepsilon[\varepsilon'/x] : \vartheta$.

Proof. By structural induction over expressions.

Case (Name). We assume that $\rho \models n : \vartheta$ and $\varepsilon' \in \rho(x)$. Since $n = n[\varepsilon'/x]$, it is immediate that also $\rho \models n[\varepsilon'/x] : \vartheta$.

Case (Public key). We assume that $\rho \models [m^+]_{id} : \vartheta$ and $\varepsilon' \in \rho(x)$. Since $[m^+]_{id} = [m^+]_{id}[\varepsilon'/x]$, it is immediate that also $\rho \models [m^+]_{id}[\varepsilon'/x] : \vartheta$.

Case (Private key). We assume that $\rho \models [m^-]_{id} : \vartheta$ and $\varepsilon' \in \rho(x)$. Since $[m^-]_{id} = [m^-]_{id}[\varepsilon'/x]$, it is immediate that also $\rho \models [m^-]_{id}[\varepsilon'/x] : \vartheta$.

Case (Variable). We assume that $\rho \models x' : \vartheta$ (therefore $\rho(x') \subseteq \vartheta$) and $\varepsilon' \in \rho(x)$. There are two cases:

1. If $\varepsilon \neq x$ then $x' = x'[\varepsilon'/x]$ and it is immediate that also $\rho \models x'[\varepsilon'/x] : \vartheta$.
2. If $\varepsilon = x$ then $x'[\varepsilon'/x] = \varepsilon'$, by hypothesis we have $\varepsilon' \in \rho(x)$ and $\rho(x') \subseteq \vartheta$, then it holds that $\rho \models \varepsilon' : \vartheta$, in which case $\rho \models x'[\varepsilon'/x] : \vartheta$.

Case (Encryption). We assume that $\rho \models \{\varepsilon_1, \dots, \varepsilon_k\}_{\varepsilon_0} : \vartheta$ and $\varepsilon' \in \rho(x)$. By the induction hypothesis it holds that $\rho \models \varepsilon_0[\varepsilon'/x] : \vartheta, \dots, \rho \models \varepsilon_k[\varepsilon'/x] : \vartheta$. Therefore, by the rule (NRAAEnc), we have $\rho \models \{\varepsilon_1, \dots, \varepsilon_k\}_{\varepsilon_0}[\varepsilon'/x] : \vartheta$.

Case (Asymmetric encryption). We assume that $\rho \models \{\mid \varepsilon_1, \dots, \varepsilon_k \mid\}^u_{\varepsilon_0}$ [from id] $: \vartheta$ and $\varepsilon' \in \rho(x)$. By the induction hypothesis it holds that $\rho \models \varepsilon_0[\varepsilon'/x] : \vartheta, \dots, \rho \models \varepsilon_k[\varepsilon'/x] : \vartheta$. Therefore, by the rule (NRAEnc), we have $\rho \models \{\mid \varepsilon_1, \dots, \varepsilon_k \mid\}^u_{\varepsilon_0}$[from id]$[\varepsilon'/x] : \vartheta$.

Since both the bases and the inductive steps have been proved, it follows that Lemma 1 holds for all the expressions by structural induction. ◻

Lemma 2 (Substitution in processes). *If* $\rho, \kappa, \psi \models \mathscr{P}$ *and* $\varepsilon \in \rho(x)$ *then* $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Proof. By structural induction over processes.

Case (Output). We assume
$\mathscr{P} = \langle \varepsilon_1, \ldots, \varepsilon_k \rangle.\mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models \langle \varepsilon_1, \ldots, \varepsilon_k \rangle.\mathscr{P}'$
- $\varepsilon \in \rho(x)$

By Lemma 1 and the induction hypothesis on the sub-processes, it holds that

- $\rho \models \varepsilon_1[\varepsilon/x] : \vartheta, \ldots, \rho \models \varepsilon_k[\varepsilon/x] : \vartheta$
- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRAOut), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Input). We assume
$\mathscr{P} = (\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k).\mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models (\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k).\mathscr{P}'$
- $\varepsilon \in \rho(x)$

By Lemma 1 and the induction hypothesis on the sub-processes, it holds that

- $\rho \models \varepsilon_1[\varepsilon/x] : \vartheta, \ldots, \rho \models \varepsilon_j[\varepsilon/x] : \vartheta$
- $\rho \models x_{j+1}[\varepsilon/x] : \vartheta, \ldots, \rho \models x_k[\varepsilon/x] : \vartheta$
- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRAInp), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Symmetric decryption). We assume
$\mathscr{P} = \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\}_{\varepsilon_0} \text{ in } \mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\}_{\varepsilon_0} \text{ in } \mathscr{P}'$
- $\varepsilon \in \rho(x)$

By Lemma 1 and the induction hypothesis on the sub-processes, it holds that

- $\rho \models \varepsilon_1[\varepsilon/x] : \vartheta, \ldots, \rho \models \varepsilon_j[\varepsilon/x] : \vartheta$
- $\rho \models x_{j+1}[\varepsilon/x] : \vartheta, \ldots, \rho \models x_k[\varepsilon/x] : \vartheta$
- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRADec), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Asymmetric decryption). We assume
$\mathscr{P} = \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k \mid\}^{u}_{\varepsilon_0} \text{ [check } NR\text{] in } \mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models \text{decrypt } \varepsilon \text{ as } \{\mid \varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k \mid\}^{u}_{\varepsilon_0} \text{ [check } NR\text{] in } \mathscr{P}'$
- $\varepsilon \in \rho(x)$

By Lemma 1 and the induction hypothesis on the sub-processes, it holds that

- $\rho \models \varepsilon_1[\varepsilon/x] : \vartheta, \ldots, \rho \models \varepsilon_j[\varepsilon/x] : \vartheta$
- $\rho \models x_{j+1}[\varepsilon/x] : \vartheta, \ldots, \rho \models x_k[\varepsilon/x] : \vartheta$
- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRAADec), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Signature). We assume
$\mathscr{P} =$ decrypt ε as $\{|\ \varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\ |\}^u_{\varepsilon_0}$ [check NR] in $\mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models$ decrypt ε as $\{|\ \varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\ |\}^u_{\varepsilon_0}$ [check NR] in $\mathscr{P}'$
- $\varepsilon \in \rho(x)$

By Lemma 1 and the induction hypothesis on the sub-processes, it holds that

- $\rho \models \varepsilon_1[\varepsilon/x] : \vartheta, \ldots, \rho \models \varepsilon_j[\varepsilon/x] : \vartheta$
- $\rho \models x_{j+1}[\varepsilon/x] : \vartheta, \ldots, \rho \models x_k[\varepsilon/x] : \vartheta$
- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRASig), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Restriction). We assume
$\mathscr{P} = (\nu\, n)\mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models (\nu\, n)\mathscr{P}'$
- $\varepsilon \in \rho(x)$

By the induction hypothesis on the sub-processes, it holds that

- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRANew), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Pair restriction). We assume
$\mathscr{P} = (\nu \pm [m]_{id})\mathscr{P}'$
By hypothesis we have

- $\rho, \kappa, \psi \models (\nu \pm [m]_{id})\mathscr{P}'$
- $\varepsilon \in \rho(x)$

By the induction hypothesis on the sub-processes, it holds that

- $\rho, \kappa, \psi \models \mathscr{P}'[\varepsilon/x]$

Therefore, by the rule (NRANew), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Parallel composition). We assume
$\mathscr{P} = \mathscr{P}_1 \mid \mathscr{P}_2$
By hypothesis we have

- $\rho, \kappa, \psi \models \mathscr{P}_1 \mid \mathscr{P}_2$

- $\varepsilon \in \rho(x)$

By the induction hypothesis on the sub-processes, it holds that

- $\rho, \kappa, \psi \models \mathscr{P}_1[\varepsilon/x]$
- $\rho, \kappa, \psi \models \mathscr{P}_2[\varepsilon/x]$

Therefore, by the rule (NRAPar), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Replication). We assume
$\mathscr{P} = [!P']_{id}$
By hypothesis we have

- $\rho, \kappa, \psi \models [!P']_{id}$
- $\varepsilon \in \rho(x)$

By the induction hypothesis on the sub-processes, it holds that

- $\rho, \kappa, \psi \models \mathscr{G}(P', id)[\varepsilon/x]$

Therefore, by the rule (NRARep), we have $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Case (Nil). We assume
$\mathscr{P} = 0$
Since $0 = 0[\varepsilon/x]$ and $\rho, \kappa, \psi \models 0$, trivially it holds $\rho, \kappa, \psi \models \mathscr{P}[\varepsilon/x]$.

Since both the basis and the inductive steps have been proved, it follows that Lemma 2 holds for all the processes by structural induction. □

Lemma 3 (Invariance of structural congruence). *If $\mathscr{P} \equiv \mathscr{Q}$ and $\rho, \kappa, \psi \models \mathscr{P}$ then $\rho, \kappa, \psi \models \mathscr{Q}$.*

Proof. By inspection of the clauses defining $\mathscr{P} \equiv \mathscr{Q}$.

Case ($\mathscr{P} \mid 0 \equiv \mathscr{P}$). We assume $\rho, \kappa, \psi \models \mathscr{P} \mid 0$, then it must be $\rho, \kappa, \psi \models \mathscr{P}$ and $\rho, \kappa, \psi \models 0$, therefore $\rho, \kappa, \psi \models \mathscr{P}$.

Other cases can be proved in a similar way, therefore Lemma 3 holds for all the clauses. □

Now, we can prove the correctness of the analysis by the Theorem defined below.

Theorem 1 (Correctness of the non-repudiation analysis). *If $\rho, \kappa, \psi \models \mathscr{P}$ and $\psi = \emptyset$ then $\mathscr{P}$ ensures* static non-repudiation.

Proof. The theorem can be proven by induction in the length of the execution sequences, showing that if $\rho, \kappa, \psi \models \mathscr{P}$ and $\mathscr{P} \rightarrow_{\mathscr{R}} \mathscr{P}'$ then $\rho, \kappa, \psi \models \mathscr{P}'$ and furthermore if $\psi = \emptyset$ then $\mathscr{P} \rightarrow_{RM} \mathscr{P}'$ does not violate the non-repudiation property.

Case (NRNCom). We assume
$\rho, \kappa, \psi \models \langle \varepsilon_1, \dots, \varepsilon_k \rangle.\mathscr{P} \mid (\varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k).\mathscr{Q}$
which amounts to:

1. $\bigwedge_{i=1}^{k} \rho \models \varepsilon_i : \vartheta_i$
2. $\forall NRV_1, \dots, NRV_k : \bigwedge_{i=1}^{k} NRV_i \propto \vartheta_i \Rightarrow \langle NRV_1, \dots, NRV_k \rangle \in \kappa$
3. $\rho, \kappa, \psi \models \mathscr{P}$
4. $\bigwedge_{i=1}^{j} \rho \models \varepsilon'_i : \vartheta'_i$
5. $\forall \langle NRV_1, \dots, NRV_k \rangle \in \kappa : \bigwedge_{i=1}^{j} NRV_i \propto \vartheta'_i \Rightarrow (\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa, \psi \models \mathscr{Q})$
6. $\bigwedge_{i=1}^{j} \varepsilon_i = \varepsilon'_i$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P} \mid \mathscr{Q}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$
From the hypothesis we obtain:

- (1.) $\Rightarrow \bigwedge_{i=1}^{k} \varepsilon_i \propto \vartheta_i$
- $\bigwedge_{i=1}^{k} \mathrm{fv}(\varepsilon_i) = \emptyset$ and (2.) $\Rightarrow \langle \varepsilon_1, \dots, \varepsilon_k \rangle \in \kappa$
- (4.) and (6.) $\Rightarrow \bigwedge_{i=1}^{j} \varepsilon_i \propto \vartheta'_i$
- (5.) $\Rightarrow \bigwedge_{i=j+1}^{k} \varepsilon_i \in \rho(\lfloor x_i \rfloor)$ and $\rho, \kappa, \psi \models \mathscr{Q}$
- Lemma 1 $\Rightarrow \rho, \kappa, \psi \models \mathscr{Q}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately
$\langle \varepsilon_1, \dots, \varepsilon_k \rangle.\mathscr{P} \mid (\varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k).\mathscr{Q} \rightarrow_{RM}$
$\mathscr{P} \mid \mathscr{Q}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$

Case (NRNDec). We assume
$\rho, \kappa, \psi \models \text{decrypt } \{\varepsilon_1, \dots, \varepsilon_k\}_{\varepsilon_0} \text{ as } \{\varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k\}_{\varepsilon'_0} \text{ in } \mathscr{P}$
which amounts to:

1. $\bigwedge_{i=0}^{k} \rho \models \varepsilon_i : \vartheta_i$
2. $\forall NRV_0, \dots, NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow \{NRV_1, \dots, NRV_k\}_{NRV_0} \propto \vartheta$
3. $\bigwedge_{i=0}^{j} \rho \models \varepsilon'_i : \vartheta'$
4. $\forall \{NRV_1, \dots, NRV_k\}_{NRV_0} \propto \vartheta : \bigwedge_{i=0}^{j} NRV_i \propto \vartheta'_i \Rightarrow (\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa, \psi \models \mathscr{P})$
5. $\bigwedge_{i=0}^{j} \varepsilon_i = \varepsilon'_i$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$
From the hypothesis we obtain:

- (1.) and $\bigwedge_{i=0}^{k} \mathrm{fv}(\varepsilon_i) = \emptyset \Rightarrow \bigwedge_{i=0}^{k} \varepsilon_i \propto \vartheta_i$
- (2.) $\Rightarrow \{\varepsilon_1, \dots, \varepsilon_k\}_{\varepsilon_0} \propto \vartheta$
- (3.) and (5.) $\Rightarrow \bigwedge_{i=0}^{j} \varepsilon_i \in \vartheta'$
- (4.) $\Rightarrow \bigwedge_{i=j+1}^{k} \varepsilon_i \in \rho(\lfloor x_i \rfloor)$ and $\rho, \kappa, \psi \models \mathscr{P}$
- Lemma 1 $\Rightarrow \rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately
decrypt $\{\varepsilon_1, \dots, \varepsilon_k\}_{\varepsilon_0}$ as $\{\varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k\}_{\varepsilon'_0}$
in $\mathscr{P} \rightarrow_{RM} \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$

Case (NRNADec). We assume
$\rho, \kappa, \psi \models$ decrypt $\{\!|\, \varepsilon_1, \dots, \varepsilon_k \,|\!\}^{u}_{\varepsilon_0}$[from id'] as
$\{\!|\, \varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k \,|\!\}^{u'}_{\varepsilon'_0}$[check NR] in $\mathscr{P}$
which amounts to:

1. $\bigwedge_{i=0}^{k} \rho \models \varepsilon_i : \vartheta_i$
2. $\forall NRV_0, \dots, NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow$
$\{\!|\, NRV_1, \dots, NRV_k \,|\!\}^{u}_{NRV_0}$[from $\lfloor id \rfloor$] $\propto \vartheta$
3. $\bigwedge_{i=0}^{j} \rho \models \varepsilon'_i : \vartheta'$
4. $\forall \{\!|\, NRV_1, \dots, NRV_k \,|\!\}_{NRV_0}$[from $\lfloor id \rfloor$] $\propto \vartheta : \forall NRV'_0 \propto \vartheta_0 : \forall([m^+]_{id'}, [m^-]_{id'}) :$
$(NRV_0, NRV'_0) = ([\lfloor m^- \rfloor]_{id'}, [\lfloor m^+ \rfloor]_{id'}) \wedge \bigwedge_{i=1}^{j} NRV_i \propto \vartheta'_i \Rightarrow (\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge$
$\rho, \kappa, \psi \models \mathscr{P})$
5. $\bigwedge_{i=1}^{j} \varepsilon_i = \varepsilon'_i$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$.
From the hypothesis we obtain:

- (1.) and $\bigwedge_{i=0}^{k} \mathrm{fv}(\varepsilon_i) = \emptyset \Rightarrow \bigwedge_{i=0}^{k} \varepsilon_i \propto \vartheta_i$
- (2.) $\Rightarrow \{\!|\, \varepsilon_1, \dots, \varepsilon_k \,|\!\}^{u}_{\varepsilon_0}$[from $\lfloor id \rfloor$] $\propto \vartheta$
- (3.) and (5.) $\Rightarrow \bigwedge_{i=0}^{j} \varepsilon_i \in \vartheta'$
- (4.) $\Rightarrow \bigwedge_{i=j+1}^{k} \varepsilon_i \in \rho(\lfloor x_i \rfloor)$ and $\rho, \kappa, \psi \models \mathscr{P}$
- Lemma 1 $\Rightarrow \rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately
decrypt $\{\!|\, \varepsilon_1, \dots, \varepsilon_k \,|\!\}^{u}_{[m^+]_{id}}$[from id'] as
$\{\!|\, \varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k \,|\!\}^{u'}_{[m^-]_{id}}$[check NR] in $\mathscr{P} \rightarrow_{RM}$
$\mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \dots, \varepsilon_k/x_k]$.

Case (NRNSig). We assume
$\rho, \kappa, \psi \models$ decrypt $\{\!|\, \varepsilon_1, \dots, \varepsilon_k \,|\!\}^{u}_{\varepsilon_0}$[from id'] as
$\{\!|\, \varepsilon'_1, \dots, \varepsilon'_j; x_{j+1}, \dots, x_k \,|\!\}^{u'}_{\varepsilon'_0}$[check NR] in $\mathscr{P}$
which amounts to:

1. $\bigwedge_{i=0}^{k} \rho \models \varepsilon_i : \vartheta_i$
2. $\forall NRV_0, \dots, NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow$
$\{\!|\, NRV_1, \dots, NRV_k \,|\!\}^{u}_{NRV_0}$[from $\lfloor id \rfloor$] $\propto \vartheta$
3. $\bigwedge_{i=0}^{j} \rho \models \varepsilon'_i : \vartheta'$
4. $\forall \{\!|\, NRV_1, \dots, NRV_k \,|\!\}^{u}_{NRV_0}$[from $\lfloor id \rfloor$] $\propto \vartheta : \forall NRV'_0 \propto \vartheta_0 : \forall([m^+]_{id'}, [m^-]_{id'}) :$
$(NRV_0, NRV'_0) = ([\lfloor m^+ \rfloor]_{id'}, [\lfloor m^- \rfloor]_{id'}) \wedge \bigwedge_{i=1}^{j} NRV_i \propto \vartheta'_i \Rightarrow (\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge$

$\rho, \kappa, \psi \models \mathscr{P} \wedge \forall nr \in NR : (id \neq id' \vee u \neq u' \vee nr \notin \{NRV_{j+1}, \ldots, NRV_k\}) \Rightarrow \lfloor nr \rfloor \in \psi)$

5. $\bigwedge_{i=1}^{j} \varepsilon_i = \varepsilon_i' \wedge RM(id, id', u, u', \varepsilon_{j+1}, \ldots, \varepsilon_k, NR)$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \ldots, \varepsilon_k/x_k]$.
From the hypothesis we obtain:

- (1.) and $\bigwedge_{i=0}^{k} \mathrm{fv}(\varepsilon_i) = \emptyset \Rightarrow \bigwedge_{i=0}^{k} \varepsilon_i \propto \vartheta_i$
- (2.) $\Rightarrow \{\mid \varepsilon_1, \ldots, \varepsilon_k \mid\}_{\varepsilon_0}^{u}[\text{from } \lfloor id \rfloor] \propto \vartheta$
- (3.) and (5.) $\Rightarrow \bigwedge_{i=0}^{j} \varepsilon_i \in \vartheta'$
- (4.) $\Rightarrow \bigwedge_{i=j+1}^{k} \varepsilon_i \in \rho(\lfloor x_i \rfloor)$ and $\rho, \kappa, \psi \models \mathscr{P}$
- Lemma 1 $\Rightarrow \rho, \kappa, \psi \models \mathscr{P}[\varepsilon_{j+1}/x_{j+1}, \ldots, \varepsilon_k/x_k]$

We observe that $\forall nr \in NR : (id \neq id' \vee u \neq u' \vee nr \notin \{\varepsilon_{j+1}, \ldots, \varepsilon_k\}) \Rightarrow \lfloor nr \rfloor \in \psi$ follows from (5.) and since $\psi = \emptyset$, must be the case that
$RM(id, id', u, u', \{\varepsilon_1, \ldots, \varepsilon_n\}, NR)$.
Thus the condition of the rule (NRNSig) are fulfilled for $\rightarrow_{RM}$.

Case (NRNNew). We assume $\rho, \kappa, \psi \models (\nu\, n)\mathscr{P}$, therefore $(\nu\, n)\mathscr{P} \rightarrow_{\mathscr{R}} (\nu\, n)\mathscr{P}'$ using rule (NRNNew) and the hypothesis $\mathscr{P} \rightarrow_{\mathscr{R}} \mathscr{P}'$.
We have to prove $\rho, \kappa, \psi \models (\nu\, n)\mathscr{P}'$.
By the induction hypothesis $\rho, \kappa, \psi \models \mathscr{P}'$ and by the rule (NRANew) $\rho, \kappa, \psi \models (\nu\, n)\mathscr{P}'$ and, when $\psi = \emptyset$, it follows immediately that $(\nu\, n)\mathscr{P} \rightarrow_{RM} (\nu\, n)\mathscr{P}'$.

Case (NRNANew). We assume $\rho, \kappa, \psi \models (\nu \pm [m]_{id})\mathscr{P}$, therefore $(\nu \pm [m]_{id})\mathscr{P} \rightarrow_{\mathscr{R}} (\nu \pm [m]_{id})\mathscr{P}'$ using rule (NRNANew) and the hypothesis $\mathscr{P} \rightarrow_{\mathscr{R}} \mathscr{P}'$.
We have to prove $\rho, \kappa, \psi \models (\nu \pm [m]_{id})\mathscr{P}'$.
By the induction hypothesis $\rho, \kappa, \psi \models \mathscr{P}'$ and by the rule (NRAANew) $\rho, \kappa, \psi \models (\nu \pm [m]_{id})\mathscr{P}'$ and, when $\psi = \emptyset$, it follows immediately that $(\nu \pm [m]_{id})\mathscr{P} \rightarrow_{RM} (\nu \pm [m]_{id})\mathscr{P}'$.

Case (NRNPar). We assume
$\rho, \kappa, \psi \models \mathscr{P}_1 \mid \mathscr{P}_2$
which amounts to:

1. $\rho, \kappa, \psi \models \mathscr{P}_1$
2. $\rho, \kappa, \psi \models \mathscr{P}_2$
3. $\mathscr{P}_1 \rightarrow_{\mathscr{R}} \mathscr{P}_1'$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P}_1' \mid \mathscr{P}_2$.
By the induction hypothesis $\rho, \kappa, \psi \models \mathscr{P}'$ and by the rule (NRAPar) $\rho, \kappa, \psi \models \mathscr{P}_1' \mid \mathscr{P}_2$ and, when $\psi = \emptyset$, it follows immediately that $\mathscr{P}_1 \mid \mathscr{P}_2 \rightarrow_{RM} \mathscr{P}_1' \mid \mathscr{P}_2$.

Case (NRNCongr). We assume
$\rho, \kappa, \psi \models \mathscr{P}$
which amounts to:

1. $\mathscr{P} \equiv \mathscr{P}^*$
2. $\mathscr{P}^* \rightarrow_{\mathscr{R}} \mathscr{P}^{**}$
3. $\mathscr{P}^{**} \equiv \mathscr{P}'$

and we have to prove
$\rho, \kappa, \psi \models \mathscr{P}'$.
By Lemma 3 and (1.) we obtain $\rho, \kappa, \psi \models \mathscr{P}^*$. By the induction hypothesis $\rho, \kappa, \psi \models \mathscr{P}^{**}$ and by Lemma 3 $\rho, \kappa, \psi \models \mathscr{P}'$ and, when $\psi = \emptyset$, it follows immediately that $\mathscr{P} \rightarrow_{RM} \mathscr{P}'$.

Case (NRNRep). We assume
$\rho, \kappa, \psi \models [!P]_{id}$
which means that $\rho, \kappa, \psi \models \mathscr{G}(P, id)$; we have to prove $\rho, \kappa, \psi \models \mathscr{G}(P, id) \mid [!P]_{id'}$. But ψ does not contain information about ids, therefore $\rho, \kappa, \psi \models \mathscr{G}(P, id^*)$ for all $id^* \in ID$, which means that $\rho, \kappa, \psi \models [!P]_{id'}$. Therefore we get $\rho, \kappa, \psi \models \mathscr{G}(P, id) \mid [!P]_{id'}$ and, when $\psi = \emptyset$, it follows immediately that $[!P]_{id} \rightarrow_{RM} \mathscr{G}(P, id) \mid [!P]_{id'}$.

Since both the basis and the inductive steps have been proved, it follows that Theorem 1 holds for all the rules by induction. □

4.3 The attacker

In the setup of $\mathscr{P} \mid \mathscr{P}_\bullet$, the attacker process $\mathscr{P}_\bullet$ has to be annotated with the extended syntax. We will use a unique label $u_\bullet$ to indicate the session and a unique label $id_\bullet$ to indicate the encryption place used by the attacker. The Dolev-Yao condition has to be redefined to be used for the non-repudiation analysis, as shown in Table 20.

The main enhancement with the usual LYSA attacker can be seen in rule (3.): whenever the attacker is able to get an encryption key and generate an encrypted message with that key, the receiver checks the *id* of the sender, and, in case the latter does not correspond to the intended one, the component ψ becomes non empty, as a signal of a non-repudiation violation.

Now we have to prove that the redefined Dolev-Yao condition holds and this is done by the following Theorem.

Theorem 2 (Correctness of Dolev-Yao Condition). *If* (ρ, κ, ψ) *satisfies* $\mathscr{F}^{DY}_{RM}$ *of type* $(\mathscr{N}_{\mathrm{f}}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ *then* $\rho, \kappa, \psi \models \overline{Q}$ *for all attackers* Q *of extended type* $(\{z_\bullet\}, \mathscr{N}_{\mathrm{f}} \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$.

Proof. By structural induction on $\overline{Q}$.

Case of (NRAOut). We assume:
$\overline{Q} = \langle \overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_k \rangle.\overline{\mathscr{P}}$
and we need to find $\vartheta_1, \ldots, \vartheta_k$ and show

Table 20 Redefinition of the attacker's capabilities

(1) The attacker may learn by eavesdropping

$$\bigwedge_{k\in\mathscr{A}_\kappa} \forall\langle NRV_1,\ldots,NRV_k\rangle \in \kappa : \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$$

(2) The attacker may learn by decrypting messages with keys already known

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{NRV_1,\ldots,NRV_k\}_{NRV_0} \in \rho(z_\bullet) : NRV_0 \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{\mid NRV_1,\ldots,NRV_k \mid\}^{u}_{[m^+]_{id}}[\text{from } id'] \in \rho(z_\bullet) : [m^-]_{id} \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall\{\mid NRV_1,\ldots,NRV_k \mid\}^{u}_{[m^-]_{id}}[\text{from } id'] \in \rho(z_\bullet) : [m^+]_{id} \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$$

(3) The attacker may construct new encryptions using the keys known

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall NRV_0,\ldots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow \{NRV_1,\ldots,NRV_k\}_{NRV_0} \in \rho(z_\bullet)$$

$$\bigwedge_{k\in\mathscr{A}_{Enc}} \forall[m^+]_{id}, NRV_1,\ldots,NRV_k : [m^+]_{id} \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow \{\mid NRV_1,\ldots,NRV_k \mid\}^{u_\bullet}_{[m^+]_{id_\bullet}} \in \rho(z_\bullet)$$

$$\begin{aligned}\bigwedge_{k\in\mathscr{A}_{Enc}} \forall[m^-]_{id}, NRV_1,\ldots,NRV_k : [m^-]_{id} \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow\\ \{\mid NRV_1,\ldots,NRV_k \mid\}^{u_\bullet}_{[m^-]_{id_\bullet}} \in \rho(z_\bullet) \wedge\\ \forall \text{ decrypt } \{\mid NRV'_1,\ldots,NRV'_k \mid\}^{u_\bullet}_{[m^-]_{id_\bullet}}[\text{from } id'] \text{ as}\\ \{\mid NRV''_1,\ldots,NRV''_j; x_{j+1},\ldots,x_k \mid\}^{u''}_{[m^+]_{id''}}[\text{check } NR] \text{ in } \mathscr{P} :\\ \forall nr \in NR\,((id' \neq id_\bullet \vee u'' \neq u_\bullet \vee\\ nr \notin \{NRV'_{j+1},\ldots,NRV'_k\}) \Rightarrow \lfloor nr \rfloor \in \psi)\end{aligned}$$

(4) The attacker may actively forge new communications

$$\bigwedge_{k\in\mathscr{A}_\kappa} \forall NRV_1,\ldots,NRV_k : \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow \langle NRV_1,\ldots,NRV_k\rangle \in \kappa$$

(5) The attacker initially has some knowledge

$\{n_\bullet, [m^\pm]_{id_\bullet}\} \cup \mathscr{N}_f \subseteq \rho(z_\bullet)$

1. $\bigwedge_{i=1}^{k} \rho \models \overline{\varepsilon}_i : \vartheta_i$
   and for all $NRV_1,\ldots,NRV_k$ with $\bigwedge_{i=1}^{k} NRV_i \propto \vartheta_i$ that
2. $\langle NRV_1,\ldots,NRV_k\rangle \in \kappa$
3. $\rho,\kappa,\psi \models \overline{\mathscr{P}}$

We choose ϑ_i $(1 \leq i \leq k)$ as the least set such that $\rho \models \overline{\varepsilon}_i : \vartheta_i$ and prove that $\vartheta_i \subseteq \rho(z_\bullet)$. If $\overline{\varepsilon}_i$ has free variables $z_1,\ldots,z_m$ then ϑ_i consists of all values $\overline{\varepsilon}_i[NRV_1/z_1,\ldots,NRV_m/z_m]$ where NRV_l $(1 \leq l \leq m) \in \rho(z_\bullet)$. This proves (1.).
(2.) is true by definition of κ.
By hypothesis, $\overline{\mathscr{P}}$ has type $(\{z_\bullet\}, \mathscr{N}_f \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ and (3.) is proved by induction hypothesis.

Case of (NRAInp). We assume:
$\overline{Q} = (\overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_j; x_{j+1}, \ldots, x_k).\overline{\mathscr{P}}$
and we need to find $\vartheta_1, \ldots, \vartheta_j$ and show

1. $\bigwedge_{i=1}^{j} \rho \models \overline{\varepsilon}_i : \vartheta_i$
   and for all $\langle NRV_1, \ldots, NRV_k \rangle \in \kappa$ with $\bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i$ that
2. $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor)$
3. $\rho, \kappa, \psi \models \overline{\mathscr{P}}$

We choose ϑ_i $(1 \leq i \leq j)$ as the least set such that $\rho \models \overline{\varepsilon}_i : \vartheta_i$ and prove that $\vartheta_i \subseteq \rho(z_\bullet)$. If $\overline{\varepsilon}_i$ has free variables $z_1, \ldots, z_m$ then ϑ_i consists of all values $\overline{\varepsilon}_i[NRV_1/z_1, \ldots, NRV_m/z_m]$ where NRV_l $(1 \leq l \leq m) \in \rho(z_\bullet)$. This proves (1.).
Since $\bigwedge_{i=1}^{j} \vartheta_i \subseteq \rho(z_\bullet)$, we have $\bigwedge_{i=1}^{j} NRV_i \in \vartheta$ and by $\mathscr{F}_{RM}^{DY}$ we get $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(z_\bullet)$ and, since $\lfloor x_i \rfloor = z_\bullet$, we have (2.).
By hypothesis, $\overline{\mathscr{P}}$ has type $(\{z_\bullet\}, \mathscr{N}_f \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ and (3.) is proved by induction hypothesis.

Case of (NRADec). We assume:
$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{\overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_j; x_{j+1}, \ldots, x_k\}_{\overline{\varepsilon}_0} \text{ in } \overline{\mathscr{P}}$
and we need to find ϑ and $\vartheta_0, \ldots, \vartheta_j$ and show

1. $\rho \models \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \overline{\varepsilon}_i : \vartheta_i$
   and for all $\{NRV_1, \ldots, NRV_k\}_{NRV_0} \propto \vartheta$ with $\bigwedge_{i=0}^{j} NRV_i \propto \vartheta_i$ that
2. $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor)$
3. $\rho, \kappa, \psi \models \overline{\mathscr{P}}$

We choose ϑ as the least set such that $\rho \models \overline{\varepsilon} : \vartheta$ and prove that $\vartheta \subseteq \rho(z_\bullet)$. If $\overline{\varepsilon}$ has free variables $z_1, \ldots, z_m$ then ϑ consists of all values $\overline{\varepsilon}[NRV_1/z_1, \ldots, NRV_m/z_m]$ where NRV_i $(1 \leq i \leq m) \in \rho(z_\bullet)$. The same development for $\vartheta_0, \ldots, \vartheta_j$ proves (1.).
Since $\vartheta_0 \subseteq \rho(z_\bullet)$, we have $NRV_0 \in \vartheta$ and by $\mathscr{F}_{RM}^{DY}$ we get $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(z_\bullet)$ and, since $\lfloor x_i \rfloor = z_\bullet$, we have (2.).
By hypothesis, $\overline{\mathscr{P}}$ has type $(\{z_\bullet\}, \mathscr{N}_f \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ and (3.) is proved by induction hypothesis.

Case of (NRAADec). We assume:
$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{\!|\, \overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_j; x_{j+1}, \ldots, x_k \,|\!\}_{\overline{\varepsilon}_0}^{u'} [\text{check } NR] \text{ in } \overline{\mathscr{P}}$
and we need to find ϑ and $\vartheta_0, \ldots, \vartheta_j$ and show

1. $\rho \models \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \overline{\varepsilon}_i : \vartheta_i$
   and for all $\{\!|\, NRV_1, \ldots, NRV_k \,|\!\}_{NRV_0}^{u} [\text{from } \lfloor id \rfloor] \propto \vartheta : \forall NRV_0' \propto \vartheta_0 : \forall (m^+, m^-) :$ $(NRV_0, NRV_0') = ([\lfloor m^- \rfloor]_{id'}, [\lfloor m^+ \rfloor]_{id'})$ with $\bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i$ that
2. $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor)$
3. $\rho, \kappa, \psi \models \overline{\mathscr{P}}$

We choose ϑ as the least set such that $\rho \models \overline{\varepsilon} : \vartheta$ and prove that $\vartheta \subseteq \rho(z_\bullet)$. If $\overline{\varepsilon}$ has free variables $z_1, \ldots, z_m$ then ϑ consists of all values $\overline{\varepsilon}[NRV_1/z_1, \ldots, NRV_m/z_m]$

where NRV_i $(1 \leq i \leq m) \in \rho(z_\bullet)$. The same development for $\vartheta_0, \ldots, \vartheta_j$ proves (1.). Since $\vartheta_0 \subseteq \rho(z_\bullet)$, we have $NRV_0 \in \vartheta$ and by $\mathscr{F}^{DY}_{RM}$ we get $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(z_\bullet)$ and, since $\lfloor x_i \rfloor = z_\bullet$, we have (2.).
By hypothesis, $\overline{\mathscr{P}}$ has type $(\{z_\bullet\}, \mathscr{N}_f \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ and (3.) is proved by induction hypothesis.

Case of (NRASig). We assume:
$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{\mid \overline{\varepsilon}_1, \ldots, \overline{\varepsilon}_j; x_{j+1}, \ldots, x_k \mid\}^{u'}_{\overline{\varepsilon}_0} [\text{check } NR] \text{ in } \overline{\mathscr{P}}$
and we need to find ϑ and $\vartheta_0, \ldots, \vartheta_j$ and show

1. $\rho \models \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models \overline{\varepsilon}_i : \vartheta_i$
   and for all $\{\mid NRV_1, \ldots, NRV_k \mid\}^{u}_{NRV_0} [\text{from } \lfloor id \rfloor] \propto \vartheta : \forall NRV_0' \propto \vartheta_0 : \forall m^+, m^-, id, id' :$
   $(NRV_0, NRV_0') = ([\lfloor m^+ \rfloor]_{id'}, [\lfloor m^- \rfloor]_{id'})$ with $\bigwedge_{i=1}^{j} NRV_i \propto \vartheta_i$ that
2. $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor)$
3. $\rho, \kappa, \psi \models \overline{\mathscr{P}}$
4. $\forall nr \in NR : (\neg RM(id, id', u, u', \{NRV_{j+1}, \ldots, NRV_k\}, \{nr\}) \Rightarrow \lfloor nr \rfloor \in \psi)$

We choose ϑ as the least set such that $\rho \models \overline{\varepsilon} : \vartheta$ and prove that $\vartheta \subseteq \rho(z_\bullet)$. If $\overline{\varepsilon}$ has free variables $z_1, \ldots, z_m$ then ϑ consists of all values $\overline{\varepsilon}[NRV_1/z_1, \ldots, NRV_m/z_m]$ where NRV_i $(1 \leq i \leq m) \in \rho(z_\bullet)$. The same development for $\vartheta_0, \ldots, \vartheta_j$ proves (1.). Since $\vartheta_0 \subseteq \rho(z_\bullet)$, we have $NRV_0 \in \vartheta$ and by $\mathscr{F}^{DY}_{RM}$ we get $\bigwedge_{i=j+1}^{k} NRV_i \in \rho(z_\bullet)$ and $\forall nr \in NR : (\neg RM(id, id', u, u', \{NRV_{j+1}, \ldots, NRV_k\}, \{nr\}) \Rightarrow \lfloor nr \rfloor \in \psi)$. Since $\lfloor x_i \rfloor = z_\bullet$, we have (2.) and (4.).
By hypothesis, $\overline{\mathscr{P}}$ has type $(\{z_\bullet\}, \mathscr{N}_f \cup \{n_\bullet\}, \mathscr{A}_\kappa, \mathscr{A}_{Enc})$ and (3.) is proved by induction hypothesis.

Case of (NRANew). We assume:
$\overline{Q} = (\nu\, n)\overline{\mathscr{P}}$
and we need to show $\rho, \kappa, \psi \models \overline{\mathscr{P}}$. But this is true by induction hypothesis.

Case of (NRAANew). We assume:
$\overline{Q} = (\nu \pm m)\overline{\mathscr{P}}$
and we need to show $\rho, \kappa, \psi \models \overline{\mathscr{P}}$. But this is true by induction hypothesis.

Case of (NRAPar). We assume:
$\overline{Q} = \overline{\mathscr{P}}_1 \mid \overline{\mathscr{P}}_2$
and we need to show

1. $\rho, \kappa, \psi \models \overline{\mathscr{P}}_1$
2. $\rho, \kappa, \psi \models \overline{\mathscr{P}}_2$

But this is true by induction hypothesis.

Case of (NRARep). We assume:
$\overline{Q} = [!\overline{P}]_{id}$
and we need to show $\rho, \kappa, \psi \models \mathscr{G}(\overline{P}, id)$. But $\mathscr{G}(\overline{P}, id)$ has the same type of $[!\overline{P}]_{id}$,

therefore $\rho, \kappa, \psi \models \mathscr{G}(\overline{P}, id)$ by induction hypothesis.

The **case (NRANil)** is trivial.

Since both the basis and the inductive steps have been proved, it follows that Theorem 2 holds for all the rules of the analysis by structural induction. □

Theorem 3. *If $\mathscr{P}$ guarantees static non-repudiation then $\mathscr{P}$ guarantees dynamic non-repudiation.*

Proof. If $\rho, \kappa, \emptyset \models \mathscr{P}_{sys}$ and $(\rho, \kappa, \emptyset)$ satisfies $\mathscr{F}^{DY}_{RM}$ then, by Theorems 1 and 2, *RM* does not abort $\mathscr{P}_{sys} \mid \overline{Q}$ regardless of the choice of attacker Q.

□

4.4 Meta Level Analysis

The analysis seen so far only deals with one session. In order to get a more realistic analysis, it has to be enhanced to a meta level, like in [3], [6]. We have to add indexes to names and variables, as explained in the Chapter 2, so a scenario with multiple principals can be modelled. The meta level non-repudiation analysis takes the form $\rho, \kappa, \psi \models_{\Gamma} M$.

Example 2. Let us now consider the protocol seen in Example 1, namely the Zhou-Gollmann protocol [12]. The whole protocol has been extended using the annotations and the functions $\mathscr{F}$ and $\mathscr{G}$. The resulting protocol is the following:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{let } X \subseteq S \text{ in } (\nu_{\pm i \in X}\ [AK_i]_{I_i})(\nu \pm\ [TTP]_{TTP})(} \\
\quad |_{i \in X}|_{j \in X} & !(\nu\ SK_{ij})(\nu\ L_{ij})(\nu\ M_{ij}) \\
& \langle f_{NRO}, I_j, L_{ij}, \{M_{ij}\}_{SK_{ij}}, \\
& \{\mid f_{NRO}, I_j, L_{ij}, \{M_{ij}\}_{SK_{ij}} \mid\}^{u_{ij}}_{[AK_i^-]_{I_i}} [\text{from } I_i]\rangle. \\
& (f_{NRR}, I_i, L_{ij}; xNRR_{ij}). \\
& \text{decrypt } xNRR_{ij} \text{ as } \{\mid f_{NRR}, I_i, L_{ij}, \{M_{ij}\}_{SK_{ij}}; \mid\}^{u_{ij}}_{[AK_j^+]_{I_j}} \\
& [\text{check } f_{NRR}, I_i, L_{ij}, \{M_{ij}\}_{SK_{ij}}] \text{ in} \\
& \langle f_{SUB}, I_j, L_{ij}, SK_{ij}, \{\mid f_{SUB}, I_j, L_{ij}, SK_{ij} \mid\}^{u_{ij}}_{[AK_i^-]_{I_i}} [\text{from } I_i]\rangle. \\
& (f_{CON}, I_i, I_j, L_{ij}, SK_{ij}; xCon_{ij}). \\
& \text{decrypt } xCon_{ij} \text{ as } \{\mid f_{CON}, I_i, I_j, L_{ij}, SK_{ij}; \mid\}^{u_{ij}}_{[TTP^+]_{TTP}} \\
& [\text{check } f_{CON}, I_i, I_j, L_{ij}, SK_{ij}] \text{ in } 0 \\
& \\
\quad ||_{i \in X}|_{j \in X} & !(f_{NRO}, I_j; xL_{ij}, xEnMsg_{ij}, xNRO_{ij}). \\
& \text{decrypt } xNRO_{ij} \text{ as } \{\mid f_{NRO}, I_j, xL_{ij}, xEnMsg_{ij}; \mid\}^{u_{ij}}_{[AK_i^+]_{I_i}} \\
& [\text{check } f_{NRO}, I_j, xL_{ij}, xEnMsg_{ij}] \text{ in}
\end{array}
$$

$$
\begin{array}{ll}
 & \langle f_{NRR}, I_i, xL_{ij}, \{\mid f_{NRR}, I_i, xL_{ij}, xEnMsg_{ij} \mid\}^{u_{ij}}_{[AK_j^-]_{I_j}} [\text{from } I_j]\rangle. \\
 & (f_{CON}, I_i, I_j, xL_{ij}; xK_{ij}, xCon_{ij}). \\
 & \text{decrypt } xCon_{ij} \text{ as } \{\mid f_{CON}, I_i, I_j, xL_{ij}, xK_{ij}; \mid\}^{u_{ij}}_{[TTP^+]_{I_{TTP}}} \\
 & [\text{check } f_{CON}, I_i, I_j, xL_{ij}, xK_{ij}] \text{ in} \\
 & \text{decrypt } xEnMsg_{ij} \text{ as } \{; xMsg_{ij}\}_{xK_{ij}} \text{ in } 0 \\
 & \\
||_{i \in X}|_{j \in X} & !(f_{SUB}, I_j; xL_{ij}, xSK_{ij}, xSub_{ij}).\text{decrypt } xSub_{ij} \text{ as} \\
 & \{\mid f_{SUB}, I_j, xL_{ij}, xSK_{ij}; \mid\}^{u_{ij}}_{[AK_i^+]_{I_i}} [\text{check } f_{SUB}, I_j, xL_{ij}, xSK_{ij}] \text{ in} \\
 & \langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \\
 & \{\mid f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij} \mid\}^{u_{ij}}_{[TTP^-]_{TTP}} [\text{from } TTP]\rangle. \\
 & \langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \\
 & \{\mid f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij} \mid\}^{u_{ij}}_{[TTP^-]_{TTP}} [\text{from } TTP]\rangle.0 \\
)
\end{array}
$$

After completing the analysis the component ψ is an empty set, i.e. the protocol guarantees non-repudiation even under attack. In fact, the attacker cannot create new encryptions because he has not knowledge about the private keys and he cannot make a replay attack because there is a unique label that identifies the session.

4.5 Over-approximation

When the analysis checks a protocol, we could expect that if the component ψ is empty then the protocol is correct, else the protocol does not guarantee the non-repudiation protocol. But the analysis cannot be precise, because of the infinitely many possible scenarios in which a protocol can be executed and the additional assumptions that can be made. Because of the over-approximation, our analysis can give sometimes a false positive, i.e. the component ψ is non empty but the protocol is correct. It is important that the analysis does not mistake in the opposite direction, and this is what happens in practice, because the analysis says that the property holds if the protocol behaves as expected, therefore it never says that a protocol is correct even if it does not guarantee the non-repudiation property. Intuitively, when a protocol guarantees authentication, freshness and integrity of the messages, it should guarantee even non-repudiation.

An example of false positive is given by the protocol described in [4] by Cederquist, Corin, and Dashti. In fact it does not use labels to identify sessions, and this is why our analysis says that this protocol does not guarantee non-repudiation property. However the protocol is correct, because it distinguishes session runs thanks to the usage of fresh keys per-session. Our analysis requires a session identifier, but there is not any element that is used in each message of the protocol, so a principal cannot verify if a message belongs to a particular session or not; indeed, without the assumption of the unique keys, an attacker could pretend to be another principal, starting the protocol after eavesdropping a protocol run. The main protocol is the following:

$$
\begin{array}{lll}
\mathrm{A} \rightarrow \mathrm{B}: & \{M\}_K, EOO_M & \text{for } EOO_M = sig_A(B, TTP, H, \{\mid K, A \mid\}_{TTP}) \\
\mathrm{B} \rightarrow \mathrm{A}: & EOR_M & \text{for } EOR_M = sig_B(EOO_M) \\
\mathrm{A} \rightarrow \mathrm{B}: & K & \\
\mathrm{B} \rightarrow \mathrm{A}: & EOR_K & \text{for } EOR_K = sig_B(A, H, K)
\end{array}
$$

where $H = h(\{M\}_K)$ and h is a hash function. There are other two sub-protocols used in case of dispute, i.e. when a principal does not finish the protocol execution, but we are interested only in the main protocol.

The encoding with annotation is the following:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{let } X \subseteq S \text{ in } (\nu_{\pm i \in X}\ [AK_i]_{I_i})(\nu \pm\ TTP)(} \\
\quad |_{i \in X}|_{j \in X} & !(\nu\ SK_{ij})(\nu\ H_{ij})(\nu\ M_{ij}) \\
& \langle \{M_{ij}\}_{SK_{ij}}, \{\mid I_j, TTP, H_{ij}, \{\mid SK_{ij}, I_i \mid\}^{u_{ij}}_{[TTP^+]_{TTP}} [\text{from } \emptyset] \mid\}^{u_{ij}}_{[AK_i^-]_{I_i}} \\
& \quad [\text{from } I_i] \rangle.(; xEORM_{ij}). \\
& \text{decrypt } xEORM_{ij} \text{ as } \{\mid \{\mid I_j, TTP, H_{ij}, \\
& \quad \{\mid SK_{ij}, I_i \mid\}^{u_{ij}}_{[TTP^+]_{TTP}} [\text{from } \emptyset] \mid\}^{u_{ij}}_{[AK_i^-]_{I_i}} [\text{from } I_i] \mid\}^{u_{ij}}_{[AK_j^+]_{I_j}} \\
& \quad [\text{check } \{\mid I_j, TTP, H_{ij}, \{\mid SK_{ij}, I_i \mid\}^{u_{ij}}_{[TTP^+]_{TTP}} [\text{from } \emptyset] \mid\}] \text{ in} \\
& \langle SK_{ij} \rangle.(; xEORK_{ij}). \\
& \text{decrypt } xEORK_{ij} \text{ as } \{\mid I_i, H_{ij}, SK_{ij}; \}^{u_{ij}}_{[AK_j^+]_{I_j}} \\
& \quad [\text{check } H_{ij}, SK_{ij}] \text{ in } 0 \\
\quad |_{i \in X}|_{j \in X} & !(; xEnMsg_{ij}, xEOOM_{ij}). \\
& \text{decrypt } xEOOM_{ij} \text{ as } \{\mid I_j, TTP; xH_{ij}, xTTP \mid\}^{u_{ij}}_{[AK_i^+]_{I_i}} \\
& \quad [\text{check } xH_{ij}] \text{ in} \\
& \langle \{\mid xEOOM_{ij} \mid\}^{u_{ij}}_{[AK_j^-]_{I_j}} [\text{from } I_j] \rangle. \\
& (; xSK_{ij}). \\
& \text{decrypt } xEnMsg_{ij} \text{ as } \{xMsg_{ij}\}_{xSK_{ij}} \text{ in} \\
& \langle \{\mid I_i, xH_{ij}, xSK_{ij} \mid\} \rangle.0 \\
)
\end{array}
$$

Because of the lack of labels, the result of the analysis shows that a possible flaw may arise. The component ψ contains all the elements that are also in NR when $|S| \geq 2$. In fact, it does not use labels to identify the session, and this is why our analysis says that this protocol does not guarantee non-repudiation property. However the protocol is correct, because of an implicit additional assumption on the uniqueness of the keys, which prevents from replay attacks.

5 Conclusions and Future Works

This paper extends the work by M. Buchholtz and H. Gao who defined a suite of analyses for security protocols, namely authentication, confidentiality, freshness, simple and complex type flaws. The annotations we introduce allow to express non-repudiation also for part of the message: this allow to tune the analysis focussing on relevant components. It results that the CFA framework developed for the process calculus LySa can be extended to security properties by identifying suitable annotations, thus re-using most of the theoretical work.

References

1. Bella, G., Paulson, L.C.: Mechanical proofs about a non-repudiation protocol. In: TPHOL01, volume 2152 of LNCS, pp. 91–104. Springer (2001)
2. Bodei, C., Buchholtz, M., Degano, P., Nielson, F., Nielson, H.R.: Static validation of security protocols. In: Journal of Computer Security, pp. 347–390 (2005)
3. Buchholtz, M., Lyngby, K.: Automated analysis of security in networking systems. ph. d. thesis proposal. available from http://www.imm.dtu.dk/ mib/thesis. Tech. rep. (2004)
4. Cederquist, Corin, Dashti: On the quest for impartiality: Design and analysis of a fair non-repudiation protocol. In: ICIS, LNCS (2005)
5. Dolev, D., Yao, A.C.: On the security of public key protocols. Tech. rep., Stanford, CA (1981)
6. Gao, H.: Analysis Of Protocols By Annotations. Ph. D. Thesis, Informatics and Mathematical Modelling, Technical University of Denmark (2008)
7. Gollmann, D.: Computer security. John Wiley & Sons, Inc., New York, NY, USA (1999)
8. Kremer, S., Raskin, J.F.: A game-based verification of non-repudiation and fair exchange protocols. In: Journal of Computer Security, pp. 551–565. Springer-Verlag (2001)
9. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer-Verlag New York, LLC (1999)
10. Schneider, S.: Formal analysis of a non-repudiation protocol. In: 11th IEEE Computer Security Foundations Workshop, p. 54 (1998)
11. Schneider, S., Holloway, R.: Security properties and csp. In: IEEE Symp. Security and Privacy, pp. 174–187. IEEE Computer Society Press (1996)
12. Zhou, J., Gollmann, D.: A fair non-repudiation protocol. IEEE Computer Society Press (1996)
13. Zhou, J., Gollmann, D.: Evidence and non-repudiation. J. Netw. Comput. Appl. **20**(3), 267–281 (1997). DOI http://dx.doi.org/10.1006/jnca.1997.0056
14. Zhou, J., Gollmann, D.: Towards verification of non-repudiation protocols. In: Proceedings of International Refinement Workshop and Formal Methods Pacific. Springer-Verlag (1998)