Computer Languages, Systems & Structures 36 (2010) 352–377

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Non-repudiation analysis using LySA with annotations

Mayla Brusò, Agostino Cortesi *

Computer Science Department, Ca' Foscari University, Via Torino, 155, Mestre, 30172 Venezia, Italy

ARTICLE INFO

Article history:
Received 16 January 2009
Received in revised form
15 April 2010
Accepted 21 April 2010

Keywords:
Static analysis
Control flow analysis
Security
Process calculus

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 Buchholtz and 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 [14–16]: 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 [12]: 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.
- The Zhou–Gollmann approach [19]: 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 Buchholtz [4] and Gao [7], 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 assigns principal identifiers and session identifiers to encryptions and decryptions.

* Corresponding author. Tel.: +39 347 4414010; fax: +39 041 2348419.
E-mail address: cortesi@dsi.unive.it (A. Cortesi).

1477-8424/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cl.2010.04.002

The advantages of this proposal are the following:

- The analysis is general enough to check any protocol (even if in few exceptional cases the result can be incorrect).
- The environment in which the protocol is executed can possibly involve infinitely many principals who run infinitely many sessions.
- The analysis can easily be implemented, providing a user-friendly tool which can automatically check the non-repudiation property for any specified encoding.

It is interesting to notice that the non-repudiation analysis that we propose easily fits into the CFA framework [13], yielding a suite of analyses that can be combined in various ways, with no major implementation overload. Since the analyses share the same framework differing only in the annotations, a combination of them might lead to a result with less resource consumption. This combination could be easily obtained by generalizing the syntax and turning the correspondent monitors on in the semantics.

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 $\pi-$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, channels are often declared private and used explicitly as cryptographic keys while in real systems they are extremely problematic. LySA calculus offers instead 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, key pairs (m^+, m^-) for asymmetric key cryptography (where m^+ is the public key and m^- is the private one), 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 $\{E_1,\ldots,E_k\}_{E_0}$ and $\{|E_1,\ldots,E_k|\}_{E_0}$, 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:

- $\langle E_1,\ldots,E_k\rangle.P$: the process sends a k-tuple of values onto the global network; when the message has been successfully sent the process continues as P.
- $(E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P$: the process reads the k-tuple of values sent, it checks if the first j values expected are identical to $E_1,\ldots,E_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $x_{j+1},\ldots,x_k$, 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 $\{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0}$ 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 $E_1,\ldots,E_j$, and, if this succeeds, the remaining $k-j$ values are bound to the variables $x_{j+1},\ldots,x_k$, 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 $\{|E_1,\ldots,E_j;x_{j+1},\ldots,x_k|\}_{E_0}$ 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.
- $P_1|P_2$: 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 Table 1.

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
$\{\|E_1,\ldots,E_k\|\}_{E_0}$	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 $\{\|E_1,\ldots,E_j;x_{j+1},\ldots,x_k\|\}_{E_0}$ in P	Asymmetric decryption
$(\nu n)P$	Restriction
$(\nu \pm m)P$	Pair restriction
$P_1\|P_2$	Parallel composition
$!P$	Replication
0	Nil

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

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

A binder introduces new names or variables which have scope in the rest of the process. The prefix (νn) 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 $x_{j+1},\ldots,x_k$. 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 Table 2 while the function $fv(P)$, defined in Table 3, collects the free variables. The bound variables are defined by the function $bv(P)\stackrel{def}{=}var(P)\backslash fv(P)$, where $var(P)$ is the function that defines the set of variables contained in a given protocol P; roughly speaking, $bv(P)$ provides the set of all the variables that are not free in the protocol P. All these functions are also defined on the terms, which are part of the processes.

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 into 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 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.

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

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

Table 4
Structural congruence $P\equiv P'$.

$P\equiv P$
$P_1\equiv P_2\Rightarrow P_2\equiv P_1$
$P_1\equiv P_2\wedge P_2\equiv P_3\Rightarrow P_1\equiv P_3$

$P_1\equiv P_2\Rightarrow$
- $\langle E_1,\ldots,E_k\rangle.P_1\equiv\langle E_1,\ldots,E_k\rangle.P_2$
- $(E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P_1 \equiv (E_1,\ldots,E_j;x_{j+1},\ldots,x_k).P_2$
- $\text{decrypt } E \text{ as } \{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0} \text{ in } P_1 \equiv \text{decrypt } E \text{ as } \{E_1,\ldots,E_j;x_{j+1},\ldots,x_k\}_{E_0} \text{ in } P_2$
- $\text{decrypt } E \text{ as } \{\|E_1,\ldots,E_j;x_{j+1},\ldots,x_k\|\}_{E_0} \text{ in } P_1 \equiv \text{decrypt } E \text{ as } \{\|E_1,\ldots,E_j;x_{j+1},\ldots,x_k\|\}_{E_0} \text{ in } P_2$
- $(\nu n)P_1\equiv(\nu n)P_2$
- $(\nu\pm m)P_1\equiv(\nu\pm m)P_2$
- $P_1|P_3\equiv P_2|P_3$
- $!P_1\equiv!P_2$

$P_1|P_2\equiv P_2|P_1$
$(P_1|P_2)|P_3\equiv P_1|(P_2|P_3)$
$P|0\equiv P$
$!P\equiv P|!P$
$(\nu n)0\equiv 0$
$(\nu n_1)(\nu n_2)P\equiv(\nu n_2)(\nu n_1)P$
$(\nu n)(P_1|P_2)\equiv P_1|(\nu n)P_2$ if $n\notin fn(P_1)$
$(\nu\pm m)0\equiv 0$
$(\nu\pm m_1)(\nu\pm m_2)P\equiv(\nu\pm m_2)(\nu\pm m_1)P$
$(\nu\pm m)(P_1|P_2)\equiv P_1|(\nu\pm m)P_2$ if $m^+,m^-\notin fn(P_1)$
$(\nu\pm m)(\nu n)P\equiv(\nu n)(\nu\pm m)P$
$P_1\stackrel{\alpha}{\equiv}P_2\Rightarrow P_1\equiv P_2$

- 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 $\alpha-$equivalent.

Two processes P_1 and P_2 are $\alpha-$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 name. The definition of the equivalence relation is in 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.

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

$$V ::= n \mid m^+$$

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

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

Table 6
Semantics of LySA calculus.

(Com)	$\dfrac{\bigwedge_{i=1}^{j} V_i = V_i'}{\langle V_1,\ldots,V_k\rangle.P \mid (V_1',\ldots,V_j'; x_{j+1},\ldots,x_k).P' \rightarrow_{\mathcal{R}} P \mid P'[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$
(Dec)	$\dfrac{\bigwedge_{i=0}^{j} V_i = V_i'}{\text{decrypt } \{V_1,\ldots,V_k\}_{V_0} \text{ as } \{V_1',\ldots,V_j'; x_{j+1},\ldots,x_k\}_{V_0'} \text{ in } P \rightarrow_{\mathcal{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$
(ADec)	$\dfrac{\bigwedge_{i=1}^{j} V_i = V_i'}{\text{decrypt } \{\mid V_1,\ldots,V_k\mid\}_{m^+} \text{ as } \{\mid V_1',\ldots,V_j'; x_{j+1},\ldots,x_k\mid\}_{m^-} \text{ in } P \rightarrow_{\mathcal{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$
(ASig)	$\dfrac{\bigwedge_{i=1}^{j} V_i = V_i'}{\text{decrypt } \{\mid V_1,\ldots,V_k\mid\}_{m^-} \text{ as } \{\mid V_1',\ldots,V_j'; x_{j+1},\ldots,x_k\mid\}_{m^+} \text{ in } P \rightarrow_{\mathcal{R}} P[V_{j+1}/x_{j+1},\ldots,V_k/x_k]}$
(New)	$\dfrac{P \rightarrow_{\mathcal{R}} P'}{(\nu n)P \rightarrow_{\mathcal{R}} (\nu n)P'}$
(ANew)	$\dfrac{P \rightarrow_{\mathcal{R}} P'}{(\nu \pm m)P \rightarrow_{\mathcal{R}} (\nu \pm m)P'}$
(Par)	$\dfrac{P_1 \rightarrow_{\mathcal{R}} P_1'}{P_1 \mid P_2 \rightarrow_{\mathcal{R}} P_1' \mid P_2}$
(Congr)	$\dfrac{P \equiv P' \wedge P' \rightarrow_{\mathcal{R}} P'' \wedge P'' \equiv P'''}{P \rightarrow_{\mathcal{R}} P'''}$

$$\mid m^-$$
$$\mid \{V_1,\ldots,V_k\}_{V_0}$$
$$\mid \{\mid V_1,\ldots,V_k\mid\}_{V_0}$$

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 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 processes. The rule (Par) is the parallel composition that can evolve into 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.

Table 7
Syntax of meta level LySA calculus.

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

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.

The meta level terms ME_i 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, so that the meta level process may instantiate to infinitely many processes, specifying arbitrarily large scenarios.
- $(v_{\bar{i}\in\bar{S}}\ n_{\overline{ai}})MP$: the process describes the restriction of all the names $n_{\overline{ai}}$; $\overline{a}$ is a prefix of the index that can be empty.
- $(v_{\pm\bar{i}\in\bar{S}}\ m_{\overline{ai}})MP$: the process describes the restriction of all the key pairs m_{ai}^{+} and m_{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; when the message has been successfully sent 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 (the one described in Section 2.1).
- 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 ME_0, then the process decrypts the k-tuple, and 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.
- decrypt ME as $\{\|ME_1,\ldots,ME_j; mx_{j+1},\ldots,mx_k\|\}_{ME_0}$ in MP: the process denotes the asymmetric decryption and it works like symmetric decryption except that ME_0 and the key used to encrypt have to be a key pair m^{+} and m^{-}.
- $(vn_{\bar{i}})MP$: the process generates k new names n_i, $i \in [1..k]$, and it continues as MP, which is the scope of the names.
- $(v \pm m_{\bar{i}})MP$: the process generates k new key pairs, m_i^{+} and m_i^{-}, and it continues as MP, which is the scope of the key pairs.
- $MP_1 | MP_2$: 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 8
Instantiation relation $MP \rightarrow_{\mathcal{I}} P$.

(ILet)	$\dfrac{MP[X \mapsto S'] \Rrightarrow P}{\text{let } X \subseteq S \text{ in } MP \Rrightarrow P}$ if $S' \subseteq_{fin} S$ where $\subseteq_{fin}$ means *finite subset*
(IIPar)	$\dfrac{MP[i \mapsto a_1] \Rrightarrow P_1 \cdots MP[i \mapsto a_k] \Rrightarrow P_k}{\mid_{i \in \{a_1,\ldots,a_k\}} MP \Rrightarrow P_1 \mid \cdots \mid P_k}$
(IINew)	$\dfrac{MP \Rrightarrow P}{(\nu_{\tilde{i} \in \{\overline{a_1},\ldots,\overline{a_k}\}}\ n_{\overline{ai}})MP \Rrightarrow (\nu n_{\overline{aa_1}}) \cdots (\nu n_{\overline{aa_k}})P}$
(IIANew)	$\dfrac{MP \Rrightarrow P}{(\nu_{\pm \tilde{i} \in \{\overline{a_1},\ldots,\overline{a_k}\}}\ m_{\overline{ai}})MP \Rrightarrow (\nu \pm\ m_{\overline{aa_1}}) \cdots (\nu m_{\overline{aa_k}})P}$
(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)	$\dfrac{MP \Rrightarrow P}{(\nu n_{\overline{a}})MP \Rrightarrow (\nu n_{\overline{a}})P}$
(IANew)	$\dfrac{MP \Rrightarrow P}{(\nu \pm m_{\overline{a}})MP \Rrightarrow (\nu \pm m_{\overline{a}})P}$
(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)	$0 \Rrightarrow 0$

An instantiation relation, written $MP \rightarrow_{\mathcal{I}} P$, 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 $\mid_{i \in S} MP$ 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.

Example 1. Let us introduce a known non-repudiation protocol, namely the Zhou–Gollmann protocol [17], 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 ↔ 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;
- $con\_K = Sig_{KTTP}(f_{CON},A,B,L,K)$ 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*);
- A→B : *X* means that principal *A* sends message *X* from principal *B*;
- A↔B : *X* means that principal *A* fetches message *X* from principal *B*.

The first message of the encoding provides *B* the encryption *C* of a message *M* under a key *K*; if the message fails to reach *B* then the protocol ends without disputes, since *B* cannot read *M* yet without the decryption key *K*. With the second step *A* is given the proof that *B* received the first message. After checking if *B*'s evidence matches with *A*'s evidence, *A* sends the decryption key *K* to the trusted third party *TTP*. Finally the trusted third party stores in a public directory a message consisting of the key and the proof that it belongs to a particular protocol session run by *A* and *B*; the principals can fetch the key through the fourth and the fifth messages (the order of the last two messages is not important).

Note that *L* and the proofs in the five messages must always match in order to eventually win a dispute, because they link the messages belonging to same session of the protocol.

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

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

where the restrictions $(v \pm KTTP)$, $(v \pm AK)$, and $(v \pm BK)$ define the key pairs used in the scope of the protocol. In particular the private keys, denoted by a minus, are used only by the subprocess modelling the behavior of the correspondent user; for example only the subprocess modelling the principal *A* can use the key AK^-. Public keys, denoted by a plus, are known by all the principals in the network so that all of them can check signatures (or encrypt messages if the protocol requires this).

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:

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

$\langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \{|f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}|\}_{KTTP^-} \rangle .$
$\langle f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}, \{|f_{CON}, I_i, I_j, xL_{ij}, xSK_{ij}|\}_{KTTP^-} \rangle .0$
)

3. Control flow analysis

In this section we introduce our Control Flow Analysis (CFA) as an extension of [13]. The aim of the CFA is to collect information about the behavior of a process and to store them in some data structures $\mathcal{A}$, 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:

$$\mathcal{A} \models P$$

which is a predicate that holds when $\mathcal{A}$ 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 $\mathcal{A}$ are such that the property they represent also holds when the process evolves. Formally:

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

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

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

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

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)	$\rho \models \{E_1, \ldots, E_k\}_{E_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0, \ldots, U_k : \bigwedge_{i=0}^{k} U_i \in \vartheta_i$ $\Rightarrow \{U_1, \ldots, U_k\}_{U_0} \in \vartheta$
(AAEnc)	$\rho \models \{\|E_1, \ldots, E_k\|\}_{E_0} : \vartheta$	iff $\bigwedge_{i=0}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_0, \ldots, U_k : \bigwedge_{i=0}^{k} U_i \in \vartheta_i$ $\Rightarrow \{\|U_1, \ldots, U_k\|\}_{U_0} \in \vartheta$
(AOut)	$\rho, \kappa \models \langle E_1, \ldots, E_k \rangle .P$	iff $\bigwedge_{i=1}^{k} \rho \models E_i : \vartheta_i \wedge \forall U_1, \ldots, U_k : \bigwedge_{i=1}^{k} U_i \in \vartheta_i$ $\Rightarrow (\langle U_1, \ldots, U_k \rangle \in \kappa \wedge \rho, \kappa \models P)$
(AInp)	$\rho, \kappa \models (E_1, \ldots, E_j; x_{j+1}, \ldots, x_k).P$	iff $\bigwedge_{i=1}^{j} \rho \models E_i : \vartheta_i \wedge \forall \langle U_1, \ldots, U_k \rangle \in \kappa : \bigwedge_{i=1}^{j} U_i \in \vartheta_i$ $\Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$
(ASDec)	$\rho, \kappa \models$ decrypt E as $\{E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{U_1, \ldots, U_k\}_{U_0} \in \vartheta$ $\wedge \bigwedge_{i=0}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$
(AADec)	$\rho, \kappa \models$ decrypt E as $\{\|E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\|\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{\|U_1, \ldots, U_k\|\}_{U_0} \in \vartheta :$ $\forall U_0' \in \vartheta_0 : \forall (m^+, m^-) : (U_0, U_0') = (\lfloor m^- \rfloor, \lfloor m^+ \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$
(AASig)	$\rho, \kappa \models$ decrypt E as $\{\|E_1, \ldots, E_j; x_{j+1}, \ldots, x_k\|\}_{E_0}$ in P	iff $\rho \models E : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \models E_i : \vartheta_i \wedge \forall \{\|U_1, \ldots, U_k\|\}_{U_0} \in \vartheta :$ $\forall U_0' \in \vartheta_0 : \forall (m^+, m^-) : (U_0, U_0') = (\lfloor m^+ \rfloor, \lfloor m^- \rfloor)$ $\wedge \bigwedge_{i=1}^{j} U_i \in \vartheta_i \Rightarrow (\bigwedge_{i=j+1}^{k} U_i \in \rho(\lfloor x_i \rfloor) \wedge \rho, \kappa \models P)$
(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)	$\rho, \kappa \models P_1 \| P_2$	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*

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

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

where $\mathcal{A}'$ 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 $\kappa \in \mathcal{P}(\lfloor Val \rfloor^*)$ collects the tuples of canonical values corresponding to the values communicated in the global network while $\rho : \lfloor Var \rfloor \to \mathcal{P}(\lfloor Val \rfloor)$ records the canonical values corresponding to the values that variables may become bound to. 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 $\vartheta \in \mathcal{P}(\lfloor Val \rfloor)$ 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 $i=1,\ldots j$; if the pattern match with the values in κ is successful, the remaining values of the k-tuple are recorded in ρ as possible binding of the variables

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)	$\rho,\kappa \models_\Gamma \mid_{i \in S} M$	iff $\bigwedge_{a \in \Gamma(S)} \rho,\kappa \models_\Gamma M[i \mapsto a]$
(MINew)	$\rho,\kappa \models_\Gamma (\nu_{\tilde{i} \in S} n_{\overline{ai}}) M$	iff $\rho,\kappa \models_\Gamma M$
(MIANew)	$\rho,\kappa \models_\Gamma (\nu_{\pm \tilde{i} \in S} m_{\overline{ai}}) M$	iff $\rho,\kappa \models_\Gamma M$
(MN)	$\rho \models n_{\tilde{i}} : \vartheta$	iff $\lfloor n_{\tilde{i}} \rfloor \in \vartheta$
(MNp)	$\rho \models m_{\tilde{i}}^+ : \vartheta$	iff $\lfloor m_{\tilde{i}}^+ \rfloor \in \vartheta$
(MNm)	$\rho \models m_{\tilde{i}}^- : \vartheta$	iff $\lfloor m_{\tilde{i}}^- \rfloor \in \vartheta$
(MVar)	$\rho \models x_{\tilde{i}} : \vartheta$	iff $\rho(\lfloor x_{\tilde{i}} \rfloor) \subseteq \vartheta$
(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*

Table 11
The attacker's capabilities.

(1)	The attacker may learn by eavesdropping
	$\bigwedge_{k\in\mathcal{A}_\kappa}\forall\langle V_1,\ldots,V_k\rangle\in\kappa:\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)$
(2)	The attacker may learn by decrypting messages with keys already known
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall\{V_1,\ldots,V_k\}_{V_0}\in\rho(z_\bullet):V_0\in\rho(z_\bullet)\Rightarrow\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall\{\lvert V_1,\ldots,V_k\rvert\}_{m^+}\in\rho(z_\bullet):m^-\in\rho(z_\bullet)\Rightarrow\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall\{\lvert V_1,\ldots,V_k\rvert\}_{m^-}\in\rho(z_\bullet):m^+\in\rho(z_\bullet)\Rightarrow\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)$
(3)	The attacker may construct new encryptions using the keys known
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall V_0,\ldots,V_k:\bigwedge_{i=0}^{k}V_i\in\rho(z_\bullet)\Rightarrow\{V_1,\ldots,V_k\}_{V_0}\in\rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall m^+,V_1,\ldots,V_k:m^+\in\rho(z_\bullet)\wedge\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)\Rightarrow\{\lvert V_1,\ldots,V_k\rvert\}_{m^+}\in\rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}}\forall m^-,V_1,\ldots,V_k:m^-\in\rho(z_\bullet)\wedge\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)\Rightarrow\{\lvert V_1,\ldots,V_k\rvert\}_{m^-}\in\rho(z_\bullet)$
(4)	The attacker may actively forge new communications
	$\bigwedge_{k\in\mathcal{A}_\kappa}\forall V_1,\ldots,V_k:\bigwedge_{i=1}^{k}V_i\in\rho(z_\bullet)\Rightarrow\langle V_1,\ldots,V_k\rangle\in\kappa$
(5)	The attacker initially has some knowledge
	$\{n_\bullet,m_\bullet^{\pm}\}\cup\mathcal{N}_f\subseteq\rho(z_\bullet)$

and the continuation process is analysed. The rules (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 $i=1,\ldots j$; if the pattern match with the values in κ is successful, the remaining values of the k-tuple are recorded in ρ as possible binding of the variables and the continuation process is analysed. Notice that the original syntax [4,7] 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 rules (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.

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

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

where $\Gamma:SetID\cup\mathcal{P}(Index_{fin})\to\mathcal{P}(Index_{fin})$ 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 $\mathcal{F}_{RM}^{DY}$ [6]. We write $P_{sys}|P_\bullet$ 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 $\mathcal{N}_f$, all the arities of the sent or received messages are in $\mathcal{A}_\kappa$ and all the arities of the encrypted or decrypted messages are in $\mathcal{A}_{Enc}$. These three sets are finite, like $\mathcal{N}_c$ and $\mathcal{X}_c$, used to collect all the names and all the variables, respectively, in the process P_{sys}. The attacker uses a new name, $n_\bullet\notin\mathcal{N}_c$, and a new variable, $z_\bullet\notin\mathcal{X}_c$, 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 $\mathcal{F}_{RM}^{DY}$ of the type $(\mathcal{N}_f,\mathcal{A}_\kappa,\mathcal{A}_{Enc})$, which is capable of characterizing the potential effect of all the attackers $P_\bullet$ of the type $(\mathcal{N}_f,\mathcal{A}_\kappa,\mathcal{A}_{Enc})$, 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 [11] 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 [18] should have verifiable origin, integrity and validity.

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 is required, as shown in Table 12. Observe that, with respect to

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
$\{\|\varepsilon_1,\ldots,\varepsilon_k\|\}^u_{\varepsilon_0}$	Asymmetric encryption
$\mathcal{P} ::=$	*Processes*
$\langle\varepsilon_1,\ldots,\varepsilon_k\rangle.\mathcal{P}$	Output
$(\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k).\mathcal{P}$	Input
decrypt ε as $\{\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\}_{\varepsilon_0}$ in $\mathcal{P}$	Symmetric decryption
decrypt ε as $\{\|\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k\|\}^u_{\varepsilon_0}$ in $\mathcal{P}$	Asymmetric decryption
$(\nu n)\mathcal{P}$	Restriction
$(\nu \pm [m]_{id})\mathcal{P}$	Pair restriction
$\mathcal{P}_1\|\mathcal{P}_2$	Parallel composition
$[!P]_{id}$	Replication
0	Nil

Table 13
Functions $\mathcal{F}$ and $\mathcal{G}$.

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

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

the LySA calculus in Table 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 $\mathcal{G}$ to the processes of the protocol analysed that acts recursively on the subprocesses and redefines subterms using another function, called $\mathcal{F}$. The definition of the functions $\mathcal{F}$ and $\mathcal{G}$, 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 asymmetric 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 have to be unique).

Notice that we have generalized the approach [7] proposed by Gao to provide freshness property in a protocol. Indeed, the author defines two functions to attach a session identifier to each statement; then, she redefines the semantics, using the functions to avoid to redefine the structural congruence. In our analysis, because of the redefinition of the latter, we do not have 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. Thus has to be removed 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:

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

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 particular session.

In practice, when there is a violation due to the *id* s, 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:

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

Furthermore, our extension involves redefinition of the semantics, of free names, of structural congruence, and of $\alpha-$equivalence, as described in 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 $\mathcal{P}$ are used instead of the previous, *E* and *P*, which do not carry annotations.
- The process $!\mathcal{P}$ is not structurally equivalent to $\mathcal{P}|!\mathcal{P}$, because of the recursive definition of the function $\mathcal{G}$.

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

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

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

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

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

$P_1 \equiv P_2 \Rightarrow [!P_1]_{id} \equiv [!P_2]_{id}$ if both P_1 and P_2 are annotated with the same id
$\mathcal{P}_1|\mathcal{P}_2 \equiv \mathcal{P}_2|\mathcal{P}_1$
$(\mathcal{P}_1|\mathcal{P}_2)|\mathcal{P}_3 \equiv \mathcal{P}_1|(\mathcal{P}_2|\mathcal{P}_3)$
$\mathcal{P}|0 \equiv \mathcal{P}$
$(\nu n)0 \equiv 0$
$(\nu n_1)(\nu n_2)\mathcal{P} \equiv (\nu n_2)(\nu n_1)\mathcal{P}$
$(\nu n)(\mathcal{P}_1|\mathcal{P}_2) \equiv \mathcal{P}_1|(\nu n)\mathcal{P}_2$ if $n \notin fn(\mathcal{P}_1)$
$(v \pm [m]_{id})0 \equiv 0$
$(v \pm [m_1]_{id})(v \pm [m_2]_{id})\mathcal{P} \equiv (v \pm [m_2]_{id})(v \pm [m_1]_{id})\mathcal{P}$
$(v \pm [m]_{id})(\mathcal{P}_1|\mathcal{P}_2) \equiv \mathcal{P}_1|(v \pm [m]_{id})\mathcal{P}_2$ if $[m^+]_{id}, [m^-]_{id} \notin fn(\mathcal{P}_1)$
$(v \pm [m]_{id})(\nu n)\mathcal{P} \equiv (\nu n)(v \pm [m]_{id})\mathcal{P}$
$\mathcal{P}_1 \overset{\alpha}{\equiv} \mathcal{P}_2 \Rightarrow \mathcal{P}_1 \equiv \mathcal{P}_2$

Table 16
Redefinition of the $\alpha-$equivalence.

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

- The rule (NRNRep) in Table 17 assures that each process has a different *id*; starting from a replication process tagged with an identifier, the rule spawns a new process with the same identifier in parallel with another replication process associated now to a fresh unique identifier id'.

We use the reference monitor semantics ($\rightarrow_{RM}$), an extension of the standard semantics ($\rightarrow_{\mathcal{R}}$), 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 $\rightarrow_{\mathcal{R}}$ checks if the requirements are met; otherwise $\mathcal{R}$ 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) ensure that the non-repudiation property holds for the elements specified by the annotations.

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

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

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

Table 17
Redefinition of the semantics of LySA calculus.

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

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)	$\rho\models \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0} : \vartheta$	iff $\bigwedge_{i=0}^{k}\rho\models\varepsilon_i : \vartheta_i \wedge \forall NRV_0,\ldots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow \{NRV_1,\ldots,NRV_k\}_{NRV_0} \propto \vartheta$
(NRAAEnc)	$\rho\models \{\mid\varepsilon_1,\ldots,\varepsilon_k\mid\}^{u}_{\varepsilon_0}[\text{from } id] : \vartheta$	iff $\bigwedge_{i=0}^{k}\rho\models\varepsilon_i : \vartheta_i \wedge \forall NRV_0,\ldots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow \{\mid NRV_1,\ldots,NRV_k\mid\}^{u}_{[NRV_0]_{id'}}[\text{from } id] \propto \vartheta$

decrypt $\{\mid NRV_1,\ldots,NRV_k\mid\}^{u}_{[m^-]_{id}}[\text{from } id']$ as
$\{\mid NRV'_1,\ldots,NRV'_j; x_{j+1},\ldots,x_k\mid\}^{u'}_{[m^+]_{id}}[\text{check } NR]$ in $\mathcal{P}$

Definition 3 says that an extended process $\mathcal{P}$ ensures non-repudiation property if there is no violation in any of its execution.

4.2. Static property

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

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 Tables 18 and 19. The rule (NRASig) checks the non-repudiation property whenever a signature is verified.

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

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

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

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 $\mathcal{P}$ is valid for every process congruent to $\mathcal{P}$.

Lemma 1 (*Substitution in expressions*). *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 $\rho \models \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0} : \vartheta$ and $\varepsilon' \in \rho(x)$. By the induction hypothesis it holds that $\rho \models \varepsilon_0[\varepsilon'/x] : \vartheta,\ldots,\rho \models \varepsilon_k[\varepsilon'/x] : \vartheta$. Therefore, by the rule (NRAAEnc), we have $\rho \models \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0}[\varepsilon'/x] : \vartheta$.

Case (Asymmetric encryption). We assume that $\rho \models \{\lvert\varepsilon_1,\ldots,\varepsilon_k\rvert\}^{u}_{\varepsilon_0}$ [from id] $: \vartheta$ and $\varepsilon' \in \rho(x)$. By the induction hypothesis it holds that $\rho \models \varepsilon_0[\varepsilon'/x] : \vartheta,\ldots,\rho \models \varepsilon_k[\varepsilon'/x] : \vartheta$. Therefore, by the rule (NRAEnc), we have $\rho \models \{\lvert\varepsilon_1,\ldots,\varepsilon_k\rvert\}^{u}_{\varepsilon_0}$[from id][ε'/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* $\rho,\kappa,\psi \vDash \mathcal{P}$ *and* $\varepsilon \in \rho(x)$ *then* $\rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon/x]$.

Proof. By structural induction over processes.

Case (Output). We assume

$$\mathcal{P} = \langle \varepsilon_1, \ldots, \varepsilon_k \rangle.\mathcal{P}'$$

By hypothesis we have

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

By Lemma 1 and the induction hypothesis on the subprocesses, it holds that

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

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

Case (Input). We assume

$$\mathcal{P} = (\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k).\mathcal{P}'$$

By hypothesis we have

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

By Lemma 1 and the induction hypothesis on the subprocesses, it holds that

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

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

Case (Symmetric decryption). We assume

$$\mathcal{P} = \text{decrypt } \varepsilon \text{ as } \{\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k\}_{\varepsilon_0} \text{ in } \mathcal{P}'$$

By hypothesis we have

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

By Lemma 1 and the induction hypothesis on the subprocesses, it holds that

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

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

Case (Asymmetric decryption). We assume

$$\mathcal{P} = \text{decrypt } \varepsilon \text{ as } \{|\varepsilon_1, \ldots, \varepsilon_j; x_{j+1}, \ldots, x_k|\}^u_{\varepsilon_0}[\text{check } NR] \text{ in } \mathcal{P}'$$

By hypothesis we have

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

By Lemma 1 and the induction hypothesis on the subprocesses, it holds that

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

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

Case (Signature). We assume

$$\mathcal{P} = \text{decrypt } \varepsilon \text{ as } \{|\varepsilon_1,\ldots,\varepsilon_j;x_{j+1},\ldots,x_k|\}^u_{\varepsilon_0}[\text{check } NR] \text{ in } \mathcal{P}'$$

By hypothesis we have

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

By Lemma 1 and the induction hypothesis on the subprocesses, it holds that

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

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

Case (Restriction). We assume

$$\mathcal{P} = (\nu n)\mathcal{P}'$$

By hypothesis we have

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

By the induction hypothesis on the subprocesses, it holds that

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

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

Case (Pair restriction). We assume

$$\mathcal{P} = (\nu \pm [m]_{id})\mathcal{P}'$$

By hypothesis we have

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

By the induction hypothesis on the subprocesses, it holds that

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

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

Case (Parallel composition). We assume

$$\mathcal{P} = \mathcal{P}_1|\mathcal{P}_2$$

By hypothesis we have

- $\rho,\kappa,\psi \vDash \mathcal{P}_1|\mathcal{P}_2$
- $\varepsilon \in \rho(x)$

By the induction hypothesis on the subprocesses, it holds that

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

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

Case (Replication). We assume

$$\mathcal{P} = [!P']_{id}$$

By hypothesis we have

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

By the induction hypothesis on the subprocesses, it holds that

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

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

Case (Nil). We assume

$$\mathcal{P} = 0$$

Since $0 = 0[\varepsilon/x]$ and $\rho,\kappa,\psi \vDash 0$, trivially it holds $\rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon/x]$.

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 $\mathcal{P} \equiv \mathcal{Q}$ and $\rho,\kappa,\psi \vDash \mathcal{P}$ then $\rho,\kappa,\psi \vDash \mathcal{Q}$.*

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

Case ($\mathcal{P}|0 \equiv \mathcal{P}$). We assume $\rho,\kappa,\psi \vDash \mathcal{P}|0$, then it must be $\rho,\kappa,\psi \vDash \mathcal{P}$ and $\rho,\kappa,\psi \vDash 0$, therefore $\rho,\kappa,\psi \vDash \mathcal{P}$.

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 $\rho,\kappa,\psi \vDash \mathcal{P}$ and $\psi = \emptyset$ then $\mathcal{P}$ ensures static non-repudiation.*

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

Case (NRNCom). We assume

$$\rho,\kappa,\psi \vDash \langle \varepsilon_1,\ldots,\varepsilon_k \rangle.\mathcal{P}|(\varepsilon'_1,\ldots,\varepsilon'_j;x_{j+1},\ldots,x_k).\mathcal{Q}$$

which amounts to:

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

and we have to prove

$$\rho,\kappa,\psi \vDash \mathcal{P}|\mathcal{Q}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$$

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) $\Rightarrow \langle \varepsilon_1,\ldots,\varepsilon_k \rangle \in \kappa$
- (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 $\rho,\kappa,\psi \vDash \mathcal{Q}$
- Lemma 1 $\Rightarrow \rho,\kappa,\psi \vDash \mathcal{Q}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately

$$\langle \varepsilon_1,\ldots,\varepsilon_k \rangle.\mathcal{P}|(\varepsilon'_1,\ldots,\varepsilon'_j;x_{j+1},\ldots,x_k).\mathcal{Q} \rightarrow_{RM} \mathcal{P}|\mathcal{Q}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$$

Case (NRNDec). We assume

$$\rho,\kappa,\psi \vDash \text{decrypt } \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0} \text{ as } \{\varepsilon'_1,\ldots,\varepsilon'_j;x_{j+1},\ldots,x_k\}_{\varepsilon'_0} \text{ in } \mathcal{P}$$

which amounts to:

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

and we have to prove

$$\rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$$

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) $\Rightarrow \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0} \propto \vartheta$
- (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 $\rho,\kappa,\psi \vDash \mathcal{P}$
- Lemma 1 $\Rightarrow \rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately

$$\begin{array}{l}\text{decrypt } \{\varepsilon_1,\ldots,\varepsilon_k\}_{\varepsilon_0} \text{ as } \{\varepsilon'_1,\ldots,\varepsilon'_j; x_{j+1},\ldots,x_k\}_{\varepsilon'_0} \\ \quad \text{in } \mathcal{P} \rightarrow_{RM} \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]\end{array}$$

Case (NRNADec). We assume

$$\begin{array}{l}\rho,\kappa,\psi \vDash \text{decrypt } \{|\varepsilon_1,\ldots,\varepsilon_k|\}^{u}_{\varepsilon_0}[\text{from } id'] \text{ as} \\ \quad \{|\varepsilon'_1,\ldots,\varepsilon'_j; x_{j+1},\ldots,x_k|\}^{u'}_{\varepsilon'_0}[\text{check } NR] \text{ in } \mathcal{P}\end{array}$$

which amounts to:

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

and we have to prove

$$\rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k].$$

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) $\Rightarrow \{|\varepsilon_1,\ldots,\varepsilon_k|\}^{u}_{\varepsilon_0}[\text{from } id] \propto \vartheta$
- (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 $\rho,\kappa,\psi \vDash \mathcal{P}$
- Lemma 1 $\Rightarrow \rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$

Therefore, when $\psi = \emptyset$, we get immediately

$$\begin{array}{l}\text{decrypt } \{|\varepsilon_1,\ldots,\varepsilon_k|\}^{u}_{[m^+]_{id}}[\text{from } id'] \text{ as} \\ \quad \{|\varepsilon'_1,\ldots,\varepsilon'_j; x_{j+1},\ldots,x_k|\}^{u'}_{[m^-]_{id}}[\text{check } NR] \text{ in } \mathcal{P} \rightarrow_{RM} \\ \quad \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k].\end{array}$$

Case (NRNSig). We assume

$$\begin{array}{l}\rho,\kappa,\psi \vDash \text{decrypt } \{|\varepsilon_1,\ldots,\varepsilon_k|\}^{u}_{\varepsilon_0}[\text{from } id'] \text{ as} \\ \quad \{|\varepsilon'_1,\ldots,\varepsilon'_j; x_{j+1},\ldots,x_k|\}^{u'}_{\varepsilon'_0}[\text{check } NR] \text{ in } \mathcal{P}\end{array}$$

which amounts to:

1. $\bigwedge_{i=0}^{k} \rho \vDash \varepsilon_i : \vartheta_i$
2. $\forall NRV_0,\ldots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \propto \vartheta_i \Rightarrow$
   $\{|NRV_1,\ldots,NRV_k|\}^{u}_{NRV_0}[\text{from } id] \propto \vartheta$
3. $\bigwedge_{i=0}^{j} \rho \vDash \varepsilon'_i : \vartheta'$
4. $\forall\{|NRV_1,\ldots,NRV_k|\}^{u}_{NRV_0}[\text{from } id] \propto \vartheta : \forall NRV'_0 \propto \vartheta_0 : \forall([m^+]_{id'},[m^-]_{id'}) :$
   $(NRV_0,NRV'_0) = ([\lfloor m^+ \rfloor]_{id'},[\lfloor m^- \rfloor]_{id'}) \wedge \bigwedge_{i=1}^{j} NRV_i \propto \vartheta'_i \Rightarrow (\bigwedge_{i=j+1}^{k} NRV_i \in \rho(\lfloor x_i \rfloor) \wedge \rho,\kappa,\psi \vDash \mathcal{P} \wedge \forall nr \in NR :$
   $(id \neq id' \vee u \neq u' \vee nr \notin \{NRV_{j+1},\ldots,NRV_k\}) \Rightarrow \lfloor nr \rfloor \in \psi)$
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

$$\rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k].$$

From the hypothesis we obtain:

- (1) and $\bigwedge_{i=0}^{k} \text{fv}(\varepsilon_i) = \emptyset \Rightarrow \bigwedge_{i=0}^{k} \varepsilon_i \propto \vartheta_i$
- (2) $\Rightarrow \{|\varepsilon_1,\ldots,\varepsilon_k|\}^{u}_{\varepsilon_0}[\text{from } id] \propto \vartheta$
- (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 $\rho,\kappa,\psi \vDash \mathcal{P}$
- Lemma 1 $\Rightarrow \rho,\kappa,\psi \vDash \mathcal{P}[\varepsilon_{j+1}/x_{j+1},\ldots,\varepsilon_k/x_k]$

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) is fulfilled for $\rightarrow_{RM}$.

Case (NRNNew). We assume $\rho,\kappa,\psi \vDash (v\ n)\mathcal{P}$, therefore $(vn)\mathcal{P} \rightarrow_{\mathcal{R}} (vn)\mathcal{P}'$ using rule (NRNNew) and the hypothesis $\mathcal{P} \rightarrow_{\mathcal{R}} \mathcal{P}'$.

We have to prove $\rho,\kappa,\psi \vDash (v\ n)\mathcal{P}'$.

By the induction hypothesis $\rho,\kappa,\psi \vDash \mathcal{P}'$ and by the rule (NRANew) $\rho,\kappa,\psi \vDash (vn)\mathcal{P}'$ and, when $\psi = \emptyset$, it follows immediately that $(vn)\mathcal{P} \rightarrow_{RM} (vn)\mathcal{P}'$.

Case (NRNANew). We assume $\rho,\kappa,\psi \vDash (v \pm [m]_{id})\mathcal{P}$, therefore $(v \pm [m]_{id})\mathcal{P} \rightarrow_{\mathcal{R}} (v \pm [m]_{id})\mathcal{P}'$ using rule (NRNANew) and the hypothesis $\mathcal{P} \rightarrow_{\mathcal{R}} \mathcal{P}'$.

We have to prove $\rho,\kappa,\psi \vDash (v \pm [m]_{id})\mathcal{P}'$.

By the induction hypothesis $\rho,\kappa,\psi \vDash \mathcal{P}'$ and by the rule (NRAANew) $\rho,\kappa,\psi \vDash (v \pm [m]_{id})\mathcal{P}'$ and, when $\psi = \emptyset$, it follows immediately that $(v \pm [m]_{id})\mathcal{P} \rightarrow_{RM} (v \pm [m]_{id})\mathcal{P}'$.

Case (NRNPar). We assume

$$\rho,\kappa,\psi \vDash \mathcal{P}_1 | \mathcal{P}_2$$

which amounts to:

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

and we have to prove

$$\rho,\kappa,\psi \vDash \mathcal{P}'_1 | \mathcal{P}_2.$$

By the induction hypothesis $\rho,\kappa,\psi \vDash \mathcal{P}'$ and by the rule (NRAPar) $\rho,\kappa,\psi \vDash \mathcal{P}'_1 | \mathcal{P}_2$ and, when $\psi = \emptyset$, it follows immediately that $\mathcal{P}_1 | \mathcal{P}_2 \rightarrow_{RM} \mathcal{P}'_1 | \mathcal{P}_2$.

Case (NRNCongr). We assume

$$\rho,\kappa,\psi \vDash \mathcal{P}$$

which amounts to:

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

and we have to prove

$$\rho,\kappa,\psi \vDash \mathcal{P}'.$$

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

Case (NRNRep). We assume

$$\rho,\kappa,\psi \vDash [!P]_{id}$$

which means that $\rho,\kappa,\psi \vDash \mathcal{G}(P,id)$; we have to prove $\rho,\kappa,\psi \vDash \mathcal{G}(P,id)|[!P]_{id'}$.

But ψ does not contain information about id s, therefore $\rho,\kappa,\psi \vDash \mathcal{G}(P,id^*)$ for all $id^* \in ID$, which means that $\rho,\kappa,\psi \vDash [!P]_{id'}$. Therefore we get $\rho,\kappa,\psi \vDash \mathcal{G}(P,id)|[!P]_{id'}$ and, when $\psi = \emptyset$, it follows immediately that $[!P]_{id} \rightarrow_{RM} \mathcal{G}(P,id)|[!P]_{id'}$.

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 $\mathcal{P}|\mathcal{P}_\bullet$, the attacker process $\mathcal{P}_\bullet$ 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 the rule (3) in Table 20: 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* $\mathcal{F}^{DY}_{RM}$ *of type* $(\mathcal{N}_f,\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ *then* $\rho,\kappa,\psi \vDash \overline{Q}$ *for all attackers* Q *of extended type* $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$.

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

Case of (NRAOut). We assume:

$$\overline{Q} = \langle \overline{\varepsilon}_1,\ldots,\overline{\varepsilon}_k \rangle.\overline{\mathcal{P}}$$

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
	$\bigwedge_{k\in\mathcal{A}_\kappa} \forall \langle NRV_1,\ldots,NRV_k \rangle \in \kappa : \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
(2)	The attacker may learn by decrypting messages with keys already known
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall \{NRV_1,\ldots,NRV_k\}_{NRV_0} \in \rho(z_\bullet) : NRV_0 \in \rho(z_\bullet) \Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall \{\lvert NRV_1,\ldots,NRV_k\rvert\}^{u}_{[m^+]_{id}}[\text{from } id'] \in \rho(z_\bullet) : [m^-]_{id} \in \rho(z_\bullet)$
	$\Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall \{\lvert NRV_1,\ldots,NRV_k\rvert\}^{u}_{[m^-]_{id}}[\text{from } id'] \in \rho(z_\bullet) : [m^+]_{id} \in \rho(z_\bullet)$
	$\Rightarrow \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
(3)	The attacker may construct new encryptions using the keys known
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall NRV_0,\ldots,NRV_k : \bigwedge_{i=0}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow \{NRV_1,\ldots,NRV_k\}_{NRV_0} \in \rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall [m^+]_{id},NRV_1,\ldots,NRV_k : [m^+]_{id} \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
	$\Rightarrow \{\lvert NRV_1,\ldots,NRV_k\rvert\}^{u_\bullet}_{[m^+]_{id_\bullet}} \in \rho(z_\bullet)$
	$\bigwedge_{k\in\mathcal{A}_{Enc}} \forall [m^-]_{id},NRV_1,\ldots,NRV_k : [m^-]_{id} \in \rho(z_\bullet) \wedge \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet)$
	$\Rightarrow \{\lvert NRV_1,\ldots,NRV_k\rvert\}^{u_\bullet}_{[m^-]_{id_\bullet}} \in \rho(z_\bullet) \wedge$
	$\forall \text{decrypt } \{\lvert NRV'_1,\ldots,NRV'_k\rvert\}^{u_\bullet}_{[m^-]_{id_\bullet}}[\text{from } id'] \text{ as}$
	$\{\lvert NRV''_1,\ldots,NRV''_j : x_{j+1},\ldots,x_k\rvert\}^{u''}_{[m^+]_{id''}}[\text{check } NR] \text{ in } \mathcal{P} :$
	$\forall nr \in NR((id' \neq id_\bullet \vee u'' \neq u_\bullet \vee$
	$nr \notin \{NRV'_{j+1},\ldots,NRV'_k\}) \Rightarrow \lfloor nr \rfloor \in \psi)$
(4)	The attacker may actively forge new communications
	$\bigwedge_{k\in\mathcal{A}_\kappa} \forall NRV_1,\ldots,NRV_k : \bigwedge_{i=1}^{k} NRV_i \in \rho(z_\bullet) \Rightarrow \langle NRV_1,\ldots,NRV_k \rangle \in \kappa$
(5)	The attacker initially has some knowledge
	$\{n_\bullet,[m^\pm]_{id_\bullet}\} \cup \mathcal{N}_f \subseteq \rho(z_\bullet)$

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

We choose $\vartheta_i (1 \le i \le k)$ as the least set such that $\rho \vDash \overline{\varepsilon}_i : \vartheta_i$ 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 $NRV_l (1 \le l \le m) \in \rho(z_\bullet)$. This proves (1).

(2) is true by definition of κ.

By hypothesis, $\overline{\mathcal{P}}$ has type $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ and (3) is proved by induction hypothesis.

Case of (NRAInp). We assume:

$$\overline{Q} = (\overline{\varepsilon}_1,\ldots,\overline{\varepsilon}_j; x_{j+1},\ldots,x_k).\overline{\mathcal{P}}$$

and we need to find $\vartheta_1,\ldots,\vartheta_j$ and show

1. $\bigwedge_{i=1}^{j} \rho \vDash \overline{\varepsilon}_i : \vartheta_i$ 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. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$

We choose $\vartheta_i (1 \le i \le j)$ as the least set such that $\rho \vDash \overline{\varepsilon}_i : \vartheta_i$ 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 $NRV_l (1 \le l \le m) \in \rho(z_\bullet)$. 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 $\mathcal{F}_{RM}^{DY}$ 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, $\overline{\mathcal{P}}$ has type $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ and (3) is proved by induction hypothesis.

Case of (NRADec). We assume:

$$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{\overline{\varepsilon}_1,\ldots,\overline{\varepsilon}_j; x_{j+1},\ldots,x_k\}_{\overline{\varepsilon}_0} \text{ in } \overline{\mathcal{P}}$$

and we need to find ϑ and $\vartheta_0,\ldots,\vartheta_j$ and show

1. $\rho \vDash \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \vDash \overline{\varepsilon}_i : \vartheta_i$ 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. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$

We choose ϑ as the least set such that $\rho \vDash \overline{\varepsilon} : \vartheta$ 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 $NRV_i (1 \le i \le m) \in \rho(z_\bullet)$. 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 $\mathcal{F}_{RM}^{DY}$ 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, $\overline{\mathcal{P}}$ has type $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ and (3) is proved by induction hypothesis.

Case of (NRAADec). We assume:

$$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{|\overline{\varepsilon}_1,\ldots,\overline{\varepsilon}_j; x_{j+1},\ldots,x_k|\}_{\overline{\varepsilon}_0}^{u'} [\text{check } NR] \text{ in } \overline{\mathcal{P}}$$

and we need to find ϑ and $\vartheta_0,\ldots,\vartheta_j$ and show

1. $\rho \vDash \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \vDash \overline{\varepsilon}_i : \vartheta_i$ and for all $\{|NRV_1,\ldots,NRV_k|\}_{NRV_0}^{u} [\text{from } id] \propto \vartheta : \forall NRV'_0 \propto \vartheta_0 : \forall (m^+,m^-) :$
   $(NRV_0,NRV'_0) = ([\lfloor m^- \rfloor]_{id'},[\lfloor m^+ \rfloor]_{id'})$ 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. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$

We choose ϑ as the least set such that $\rho \vDash \overline{\varepsilon} : \vartheta$ 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 $NRV_i (1 \le i \le m) \in \rho(z_\bullet)$. 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 $\mathcal{F}_{RM}^{DY}$ 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, $\overline{\mathcal{P}}$ has type $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ and (3) is proved by induction hypothesis.

Case of (NRASig). We assume:

$$\overline{Q} = \text{decrypt } \overline{\varepsilon} \text{ as } \{|\overline{\varepsilon}_1,\ldots,\overline{\varepsilon}_j; x_{j+1},\ldots,x_k|\}_{\overline{\varepsilon}_0}^{u'} [\text{check } NR] \text{ in } \overline{\mathcal{P}}$$

and we need to find ϑ and $\vartheta_0,\ldots,\vartheta_j$ and show

1. $\rho \vDash \overline{\varepsilon} : \vartheta \wedge \bigwedge_{i=0}^{j} \rho \vDash \overline{\varepsilon}_i : \vartheta_i$ and for all $\{|NRV_1,\ldots,NRV_k|\}_{NRV_0}^{u} [\text{from } id] \propto \vartheta : \forall NRV'_0 \propto \vartheta_0 : \forall m^+,m^-,id,id' :$
   $(NRV_0,NRV'_0) = ([\lfloor m^+ \rfloor]_{id'},[\lfloor m^- \rfloor]_{id'})$ 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. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$
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 $\rho \vDash \overline{\varepsilon} : \vartheta$ 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 $NRV_i (1 \le i \le m) \in \rho(z_\bullet)$. 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 $\mathcal{F}^{DY}_{RM}$ 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, $\overline{\mathcal{P}}$ has type $(\{z_\bullet\},\mathcal{N}_f \cup \{n_\bullet\},\mathcal{A}_\kappa,\mathcal{A}_{Enc})$ and (3) is proved by induction hypothesis.

Case of (NRANew). We assume:

$$\overline{Q} = (\nu n)\overline{\mathcal{P}}$$

and we need to show $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$. But this is true by induction hypothesis.

Case of (NRAANew). We assume:

$$\overline{Q} = (\nu \pm m)\overline{\mathcal{P}}$$

and we need to show $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}$. But this is true by induction hypothesis.

Case of (NRAPar). We assume:

$$\overline{Q} = \overline{\mathcal{P}}_1 | \overline{\mathcal{P}}_2$$

and we need to show

1. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}_1$
2. $\rho,\kappa,\psi \vDash \overline{\mathcal{P}}_2$

But this is true by induction hypothesis.

Case of (NRARep). We assume:

$$\overline{Q} = [!\overline{P}]_{id}$$

and we need to show $\rho,\kappa,\psi \vDash \mathcal{G}(\overline{P},id)$. But $\mathcal{G}(\overline{P},id)$ has the same type of $[!\overline{P}]_{id}$, therefore $\rho,\kappa,\psi \vDash \mathcal{G}(\overline{P},id)$ 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 $\mathcal{P}$ guarantees static non-repudiation then $\mathcal{P}$ guarantees dynamic non-repudiation.*

Proof. If $\rho,\kappa,\emptyset \vDash \mathcal{P}_{sys}$ and $(\rho,\kappa,\emptyset)$ satisfies $\mathcal{F}^{DY}_{RM}$ then, by Theorems 1 and 2, *RM* does not abort $\mathcal{P}_{sys}|\overline{Q}$ 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 [4,7]. We have to add indexes to names and variables, as explained in Section 2, so a scenario with multiple principals can be modelled. The meta level non-repudiation analysis takes the form $\rho,\kappa,\psi \vDash_\Gamma M$.

Example 2. Let us now consider the protocol seen in Example 1, namely the Zhou–Gollmann protocol [17]. The whole protocol has been extended using the annotations and the functions $\mathcal{F}$ and $\mathcal{G}$. The resulting protocol is the following:

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

$\langle f_{NRR},I_i,xL_{ij},\{|f_{NRR},I_i,xL_{ij},xEnMsg_{ij}|\}^{u_{ij}}_{[AK_j^-]_{I_j}}$ [from I_j]$\rangle$.
($f_{CON},I_i,I_j,xL_{ij};xK_{ij},xCon_{ij}$).
decrypt $xCon_{ij}$ as $\{|f_{CON},I_i,I_j,xL_{ij},xK_{ij};|\}^{u_{ij}}_{[KTTP^+]_{TTP}}$
[check $f_{CON},I_i,I_j,xL_{ij},xK_{ij}$] in
decrypt $xEnMsg_{ij}$ as $\{;xMsg_{ij}\}_{xK_{ij}}$ in 0

$\|_{i\in X}|_{j\in X}$ $!(f_{SUB},I_j;xL_{ij},xSK_{ij},xSub_{ij})$.decrypt $xSub_{ij}$ as
$\{|f_{SUB},I_j,xL_{ij},xSK_{ij};|\}^{u_{ij}}_{[AK_i^+]_{I_i}}$ [check $f_{SUB},I_j,xL_{ij},xSK_{ij}$] in
$\langle f_{CON},I_i,I_j,xL_{ij},xSK_{ij}$,
$\{|f_{CON},I_i,I_j,xL_{ij},xSK_{ij}|\}^{u_{ij}}_{[KTTP^-]_{TTP}}$[from TTP]$\rangle$.
$\langle f_{CON},I_i,I_j,xL_{ij},xSK_{ij}$,
$\{|f_{CON},I_i,I_j,xL_{ij},xSK_{ij}|\}^{u_{ij}}_{[KTTP^-]_{TTP}}$[from TTP]$\rangle$.0

)

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 [5] by Cederquist et al. 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:

A→B : $\{M\}_K$, EOO_M for $EOO_M = sig_A(B,TTP,H,\{|K,A|\}_{TTP})$
B→A : EOR_M for $EOR_M = sig_B(EOO_M)$
A→B : K
B→A : EOR_K for $EOR_K = sig_B(A, H, K)$

where $H=h(\{M\}_K)$ and h is a hash function. There are other two subprotocols 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:

let $X \subseteq S$ in $(\nu_{\pm i\in X}[AK_i]_{I_i})(\nu \pm KTTP)($
$|_{i\in X}|_{j\in X}$ $!(\nu SK_{ij})(\nu H_{ij})(\nu M_{ij})$
$\langle \{M_{ij}\}_{SK_{ij}},\{|I_j,TTP,H_{ij},\{|SK_{ij},I_i|\}^{u_{ij}}_{[KTTP^+]_{TTP}}$[from TTP]$|\}^{u_{ij}}_{[AK_i^-]_{I_i}}$
[from I_i]$\rangle$.(;$xEORM_{ij}$).
decrypt $xEORM_{ij}$ as $\{|\{|I_j,TTP,H_{ij},$
$\{|SK_{ij},I_i|\}^{u_{ij}}_{[KTTP^+]_{TTP}}$[from TTP]$|\}^{u_{ij}}_{[AK_i^-]_{I_i}}$[from I_i]$|\}^{u_{ij}}_{[AK_j^+]_{I_j}}$
[check $\{|I_j,TTP,H_{ij},\{|SK_{ij},I_i|\}^{u_{ij}}_{[KTTP^+]_{TTP}}$[from TTP]$|\}$] in
$\langle SK_{ij}\rangle$.(;$xEORK_{ij}$).
decrypt $xEORK_{ij}$ as $\{|I_i,H_{ij},SK_{ij};\}^{u_{ij}}_{[AK_j^+]_{I_j}}$
[check H_{ij},SK_{ij}] in 0

$|_{i\in X}|_{j\in X}$ $!(;xEnMsg_{ij},xEOOM_{ij})$.
decrypt $xEOOM_{ij}$ as $\{|I_j,TTP;xH_{ij},xTTP|\}^{u_{ij}}_{[AK_i^+]_{I_i}}$
[check xH_{ij}] in
$\langle \{|xEOOM_{ij}|\}^{u_{ij}}_{[AK_j^-]_{I_j}}$[from I_j]$\rangle$.
(;xSK_{ij}).

decrypt $xEnMsg_{ij}$ as $\{xMsg_{ij}\}_{xSK_{ij}}$ in
$\langle\{|I_i,xH_{ij},xSK_{ij}|\}\rangle.0$

)

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 Buchholtz and Gao who defined a suite of analyses for security protocols, namely authentication, confidentiality [10], freshness [9], simple [3] and complex [8] type flaws. The annotations we introduce allow to express non-repudiation also for part of the message: this allows 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.

Acknowledgments

Work partially supported by MIUR Project PRIN 2007 "SOFT" and by RAS Project TESLA—Tecniche di enforcement per la sicurezza dei linguaggi e delle applicazioni.

References

[1] Bella G, Paulson LC. Mechanical proofs about a non-repudiation protocol. In: TPHOL'01, Lecture notes in computer science, vol. 2152. Berlin, Germany: Springer; 2001. p. 91–104.
[2] Bodei C, Buchholtz M, Degano P, Nielson F, Nielson HR. Static validation of security protocols. J Comput Security 2005;347–90.
[3] Bodei C, Degano P, Gao H, Brodo L. Detecting and preventing type flaws: a control flow analysis with tags. In: Proceedings of 5th international workshop on security issues in concurrency, Lisboa, Portugal, 2007.
[4] Buchholtz M, Lyngby K. Automated analysis of security in networking systems. PhD thesis proposal. Available from ⟨http://www.imm.dtu.dk/mib/thesis⟩. Technical Report; 2004.
[5] Cederquist J, Corin R, Torabi Dashti M. On the quest for impartiality: design and analysis of a fair non-repudiation protocol. In: ICIS, Lecture notes in computer science, 2005.
[6] Dolev D, Yao AC. On the security of public key protocols. Technical Report, Stanford, CA; 1981.
[7] Gao H. Analysis of protocols by annotations. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark; 2008.
[8] Gao H, Bodei C, Degano, P. A formal analysis of complex type flaw attacks on security protocols. In: Proceeding of 12th international conference on algebraic methodology and software technology (AMAST'08), USA, 2008.
[9] Gao H, Degano P, Bodei C, Nielson HR. A formal analysis for capturing replay attacks in cryptographic protocols. In: Proceeding of 12th annual asian computing science conference: focusing on computer and network security (ASIAN'07), Doha, Qatar, 2007.
[10] Gao H, Nielson HR. Analysis of lysa-calculus with explicit confidentiality annotations. In: Proceedings of 20th international conference on advanced information networking and applications (AINA 2006), Vienna, Austria. Los Alamitos, CA, USA: IEEE Computer Society; 2006.
[11] Gollmann D. Computer security. New York, NY, USA: John Wiley & Sons, Inc.; 1999.
[12] Kremer S, Raskin JF. A game-based verification of non-repudiation and fair exchange protocols. J Comput Security 2001:551–65.
[13] Nielson F, Nielson HR, Hankin C. Principles of program analysis. New York: Springer-Verlag, LLC; 1999.
[14] Schneider S. Formal analysis of a non-repudiation protocol. In: 11th IEEE computer security foundations workshop, 1998. p. 54.
[15] Schneider S, Holloway R. Modelling security properties with csp. Technical Report; 1996.
[16] Schneider S, Holloway R. Security properties and csp. In: IEEE symposium security and privacy. Los Alamitos, CA, USA: IEEE Computer Society Press; 1996. p. 174–87.
[17] Zhou J, Gollmann D. A fair non-repudiation protocol. IEEE Computer Society Press; 1996.
[18] Zhou J, Gollmann D. Evidence and non-repudiation. J Network Comput Appl 1997;20(3):267–81 doi:http://dx.doi.org/10.1006/jnca.1997.0056.
[19] Zhou J, Gollmann D. Towards verification of non-repudiation protocols. In: Proceedings of international refinement workshop and formal methods pacific. Berlin, Germany: Springer-Verlag; 1998.