A Secure Key Management Interface with Asymmetric Cryptography

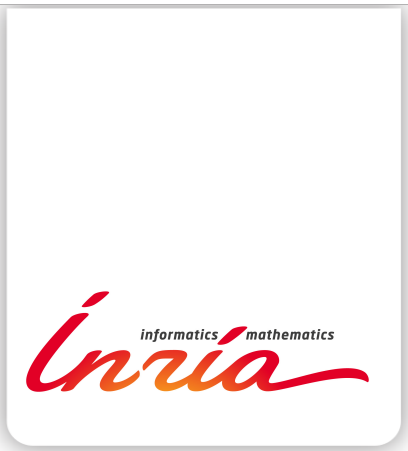
Marion Daubignard, David Lubicz, Graham Steel

A Secure Key Management Interface with Asymmetric Cryptography

Marion Daubignard, David Lubicz, Graham Steel

RESEARCH REPORT
N° 8274
April 2013
Project-Team Prosecco

ISSN 0249-6399 ISRN INRIA/RR--8274--FR+ENG



A Secure Key Management Interface with Asymmetric Cryptography

Marion Daubignard*, David Lubicz*, Graham Steel

Project-Team Prosecco

Research Report n° 8274 — April 2013 — 26 pages

* DGA.MI, BP 57419 35174 Bruz Cedex, France

RESEARCH CENTRE
PARIS – ROCQUENCOURT

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Abstract: Cryptographic devices such as Hardware Security Modules are only as secure as their application programme interfaces (APIs) that offer cryptographic functionality to the outside world. Design flaws and implementation errors in security APIs have been shown to cause vulnerabilities that may leak secrets such as keys and PINs. Ideally, we would like to design such interfaces in such a way that we can formally prove security properties, even in the presence of some corrupted keys. In this work, we take such a design for a provably secure interface for symmetric key management, due to Cortier and Steel, and extend it to asymmetric cryptography, giving new security definitions and associated proofs. Asymmetric cryptography forces us to consider confidentiality and integrity properties separately and provide support for classical operations of public key infrastructure (e.g. certification of public keys). As far as we are aware this is the first such provably secure interface to support asymmetric key operations for key management: Cachin and Chandran's secure token interface supports asymmetric key operations only for encrypting and signing data, not for managing keys.

Key-words: crytography, key management, security APIs

Une généralisation de l'API Cortier-Steel pour la cryptographie asymétrique

Résumé : Les systèmes cryptographiques tels que les modules matériels de sécurité ne peuvent apporter un niveau de sécurité que dans la mesure où leur interface de programmation (API) qui offre les services de cryptographie à l'extérieur de module atteint ce même niveau de sécurité. Il a été constaté que des défauts de conception ou des erreur d'implémentation dans les APIs de sécurité sont à l'origine de vulnérabilités pouvant entrainer la fuite de secrets comme des clefs ou des PINs. Idéalement, nous voudrions concevoir de telles interfaces de manière à pouvoir prouver formellement des propriétés de sécurité, même si certaines clefs sont corrompues. Dans cet article, nous partons d'une telle API, due à Cortier et Steel, conçue de manière disposer d'une preuve de sécurité pour la gestion de clefs symétriques, et nous l'adaptons à la cryptographie asymétrique en donnant une nouvelle définition de sécurité avec les preuves associées. Afin de prendre en compte la cryptographie asymétrique, nous sommes amenés à gérer de manière différentiée les propriétés de confidentialité et d'intégrité et à ajouter les fonctionalités classiques d'une infrastructure de gestion de clefs publiques (i.e. la certification des clefs publiques). Á notre connaissance, il s'agit de la première preuve d'interface prouvée permettant l'usage de primitives asymétriques pour la gestion de clefs : l'interface de Cachin et Chandra prévoit l'usage de primitives asymétriques uniquement pour le chiffrement et la signature de données, et non pas pour la gestion des clefs.

Mots-clés : cryptographie, gestion des clés, sécurité des APIs

1 Introduction

In a context of constant security threats combined with increasing heterogeneity of platforms and applications, developers are turning more and more to solutions based on secure hardware, whether it be a smartcard, Trusted Platform Module (TPM), hardware-secured virtual execution environment (e.g. TrustZone) or Hardware Security Module (HSM). In a typical architecture, the secure hardware contains cryptographic keys and the ability to perform some basic crypto operations which can be leveraged to ensure security for the whole system. However, designing the application programme interface (API) of such a device is difficult: it must allow the user to manage the keys on the device and access the crypto without allowing an attacker, who may in the worse case be able to make arbitrary calls to the API, to be able to obtain secrets. Many attacks have been found on the APIs of contemporary devices []. One promising approach to solving this problem is to design APIs such that one can formally prove security properties in the presence of a suitably powerful intruder. Such an approach has been applied both in the standard cryptographic model [2] and the symbolic or Dolev-Yao model [4].

However, neither of these designs present a scheme for managing keys using asymmetric cryptography, which is often used in practice for such a task since it provides a convenient way to bootstrap security without any pre-shared secrets. In this work, we extend the API designed by Cortier and Steel [] to include key management with asymmetric cryptography. To do so, we have to add an explicit mechanism for assuring the integrity of keys to be imported onto a device, since anyone can potentially encrypt using a public key leading to so-called "Trojan key" attacks [3]. We add signature keys for signing encryption under public keys and also separate certification keys, the latter used to manage the public key infrastructure (PKI) of keys and certificates. As far as we aware this is the first such design to be proposed with security proofs.

In the rest of the paper, we will first introduce our symbolic model and explain the features of our API design 2. We describe the API rules formally 3, and then give the security properties and sketch proofs 4. Full proofs are given in section 5.

Related Work The API we are building on was proposed by Cortier and Steel [4]. It supports only symmetric key cryptography, but can nonetheless be used to implement any secure symmetric key exchange protocol from the Clark-Jacob corpus. The main principle is that keys are arranged in a hierarchy of levels. Each key is associated to its level and the set of agents who are allowed to use it. This association is made when storing the key on the device, by including it as metadata stored with the key, and when encrypting the key for transfer, by tagging the encrypted key with exactly this information. The API rules are designed such that keys may only be encrypted by other keys which are higher in the hierarchy, i.e. they are at least one level higher and assigned to a set of agents that is equal to or smaller than the payload key. We will generalise this notion slightly in our API. The Cortier and Steel (CS) API also includes a notion of freshness for imported keys enforced by nonces. Although we do not include that mechanisms in our extension, it should be simple to add if needed.

Cachin and Chandran proposed an API with a quite different design [2]. They rely on the fact that all keys are stored on a central key server. Instead of assigning security attributes such as levels and agent identifiers to keys at creation time, they allow the key's role to evolve over time by logging all operations, and then disallowing operations that would be insecure by observing the log. They allow asymmetric keys to be managed by symmetric key cryptography, but do not allow asymmetric keys to be used for key management operations like export and import.

Other work has investigated the foundations of models for secure key management APIs: Kremer, Steel and Warinschi give a model that can be interpreted in the symbolic and computational cryptography worlds [6]. They show that the possibility of key corruption requires strong assumptions to be made on the key wrapping primitives in the computational model. Recent work by Künnemann, Kremer and Steel investigates composable notions of security for key management [5]. This is an ap-

pealing idea because it allows (almost) arbitrary secure cryptographic primitives to be used with the keys under management without having to repeat the security proofs, but currently only management with symmetric keys is supported.

2 Design of the API

We first describe the roles assigned to keys in our API. We then give the syntax and informal semantics for the message algebra and introduce our notion of *key handles* which extends previous designs.

2.1 Key Types

The original Cortier-Steel API was limited to symmetric keys management. In this paper, we present an extension to deal with asymmetric keys. In order to limit the number of key roles that we introduce, we consider that the asymmetric keys are double keys: it means that the same key can be used as an input of both an encryption and a signature scheme. Thus, we have encryption/verification public keys and decryption/signature private keys. It is clear that in practise a double key can simply be obtained by the concatenation of a signature and encryption key and that a usual simple key can be simulated by a double key. Thus, we do not lose generality with this simplification. On the other side, it makes sense from a security point of view since the encrypt and sign operation is the minimal basic operation which ensure the confidentiality of message and an authentication of the issuer which is mandatory for the set up of our security policy. Signature keys are used to sign encryptions of other keys or messages. Asymmetric public keys are certified by certification keys (with a signature algorithm).

The list of key roles that we are going to manipulate is:

- symmetric encryption/decryption keys;
- encryption/verification of signature double public keys;
- decryption/signature double private keys;
- verification of certificates public keys;
- certification private keys.

It is possible that the algorithm used to sign the certificates is the same as the algorithm to sign the encrypted messages. Nonetheless, for security reasons, it is important to distinguish the key roles to privent a signature algorithm from being used as a certification oracle by an adversary. The different key roles and their associated key types are summarised in the table 1. We denoted by $\mathcal{T}$ the set of key types.

Key	Role	Type
Priv	decryption/signature private key	privDecSign
Pub	encryption/verification of signature public key	pubEncVerif
Sym	symmetric encryption key	symEncDec
pub	certificate verification key	pubCertVerif
priv	certificate signature key	privCertSign

Figure 1: Table of the set of key roles and types ($\mathcal{T}$)

2.2 Security Levels

The set of key security levels I is supposed to a finite set together with an partial strict order relation denoted $<$. We suppose that there is a minimal element in I denoted by 0. By definition, for all $x \in I \setminus 0$, we have $0 < x$. The 0 element represents the security level of public information. We are given a partition of I in two subsets :

- the levels $I_1 \subset I$ which correspond to the keys which can only encrypt regular messages ;
- the levels $I_2 \subset I$ which correspond to the keys which can be used to transport keys of level I_1.

We set $I_{>0} = I_1 \cup I_2 = I - \{0\}$.

2.3 Message Algebra

The messages are represented by a term algebra. We suppose given an infinite set of agents Agent, an infinite set of nonces Nonce and an infinite set of keys Key. We are also given an infinite set of variables Var in which we distinguish an infinite set of key variables VarKey and an infinite set of nonce variables VarNonce. The term algebra is given by :

$$\begin{array}{rcl} \mathsf{Keyv} & ::= & \mathsf{Key} \mid \mathsf{VarKey} \mid inv(\mathsf{Keyv}) \\ \mathsf{Noncev} & ::= & \mathsf{Nonce} \mid \mathsf{VarNonce} \\ \mathsf{Msg} & ::= & \mathsf{Agent} \mid \mathsf{Keyv} \mid \mathsf{Noncev} \mid I \mid \mathcal{T} \mid \{\!|\mathsf{Msg}|\!\}_{\mathsf{Keyv}} \mid \{\mathsf{Msg}\}_{\mathsf{Keyv}} \\ & & \mid \Sigma(\mathsf{Msg}, \mathsf{Keyv}) \mid \mathsf{nhdl}(\mathsf{Msg}) \mid < \mathsf{Msg}, \mathsf{Msg} > \\ \mathsf{Handle} & ::= & h^{\alpha}_{\mathsf{Agent}}(\mathsf{Noncev}, \mathsf{Noncev}, \mathsf{Msg}, \mathcal{T}, I, \mathscr{S}, \mathscr{S}) \\ & & \mid h_{\mathsf{Agent}}(\mathsf{Noncev}, \mathsf{Msg}) \end{array}$$

where $\mathscr{S}$ is the set of sub-sets of Agent.

The set Keyv represents the set of keys and variable of keys. A term of the form $inv(k)$ with $k \in \mathsf{Key}$ represents the private key associated to the public key k. The set Noncev is the set of nonces and variable of nonces. The terms of type Msg are made of elements of Agent, Keyv, Noncev together with constructors representing encryption, signature together with sets needed to represent the attributes of the handles. More precisely,

- the term $\{\!|m|\!\}_k$ represents the symmetric encryption of the message m with the key k;
- the term $\{m\}_k$ represents the asymmetric encryption of the message m with the double key k;
- the term $\Sigma(m, k)$ represents the signature of the message m with the double key k ;
- the term nhdl() allows one to encapsulate a regular message which does not correspond to the transportation of a handle (see below) ;
- the term $< m_1, m_2 >$ represents the pair of the two messages $m_1, m_2 \in \mathsf{Msg}$.

For $n > 0$ an integer, we will often write $m_1, \ldots, m_n$ in place of $< m_1, < m_2, < \ldots, m_n >>>$ in order to simplify the notations.

2.4 Handles

The API does not give direct access to the data stored on the HSM. Instead an agent can manipulate the data by the way of handles. We have two types of handle :

- *key handles* are used to protect integrity and confidentiality of the data on the HSM
- *integrity handles* used to protect the integrity of data the HSM.

The key handles are terms of the form $h_a^{\alpha}(N_1, N_2, m, T, i, S_1, S_2)$, corresponding to the following information :

- the agent $a \in \mathsf{Agent}$ which own the handle ;
- the identifier $N_1 \in \mathsf{Nonce}$ (unique in the whole system) of the handle ;
- if m is a double private key, N_2 is the identifier of the associated certificate of the double public key, else $N_2 = \mathsf{Null}$;
- the message $m \in \mathsf{Msg}$ (usually m is a key or a nonce) associated to the handle ;
- le type $T \in \mathcal{T}$ of the message (see table 1 for a list of possible types) ;
- the triple $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$ is the security level of the handle (the security policy of the API is based on this structure);
- the label $\alpha \in \{r, g\}$ allows to distinguish the keys which have been generated by A ($\alpha = g$) from the keys which have been received and imported ($\alpha = r$).

The integrity handles are terms of the form $h_a(N_1, m)$ with an identifier N_1 and a message $\mathsf{m} \in \mathsf{Msg}$. They are meant to model the preservation of the integrity of data by a signature : if the signature of a message m is valid then the API produces an integrity handle containing the message m. Contrarily to the common usage in cryptography where a public key certificate corresponds to signed public informations, we distinguish here two elements, the pre-certificate and the certificate which is a signed pre-certificate. Indeed, the outcome of the certificate verification operation is a new pre-certificate stored under an integrity handle in the HSM.

We remark that we have included in the data structure of a handle the identifier of the associated public key pre-certificate because we want the API to manage pairs of associated public/private keys. As a consequence, we need the pre-certificate to have an identifier which is independent of HSM it is stored into. This explains why the pre-certificate contains a field corresponding to its identifier (which can not be decided randomly by the HSM when the pre-certificate is imported).

In the following, for clarity, we will use the notation $\mathcal{C}(N_1, N_2, N_3, k, T, i, S_1, S_2)$, which is a synonym of the concatenation of the terms $N_1, N_2, N_3, k, T, i, S_1, S_2 \in \mathsf{Msg}$, to represent a certificate of double public key. It is important to note that in the notation $\mathcal{C}(N_1, N_2, N_3, k, T, i, S_1, S_2)$, we do not make hypothesis on the type of the fields. Nonetheless, we will say that a pre-certificate is *well formed* if the fields it contains correspond to the following terms and types (we also give their semantic) :

- the identifier $N_1 \in \mathsf{Nonce}$ of the certificate;
- the identifier $N_2 \in \mathsf{Nonce}$ of the associated private key;
- the identifier $N_3 \in \mathsf{Nonce}$ of the certification public key which allows to verify the certificate;
- a double public key $k \in \mathsf{Key}$;
- the type $T \in \mathcal{T}$ of k ;
- the associated private key handle security level $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$.

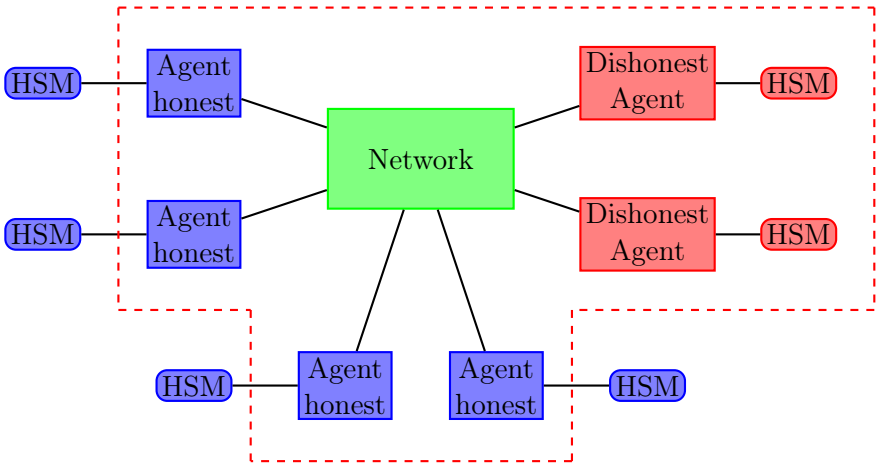


Figure 2: Corruption model

2.5 API Rules

The API is going to be represented by a set of rules which use variables. These rules will be instantiated by substituting the variables by terms of the same type. In order to explain that, let $x_1, \ldots, x_n$ be elements of Var and let $t_1, \ldots, t_n$ be a set of terms. We denote by $\{x_1 \to t_1, \ldots, x_n \to t_n\}$ the substitution σ which replaces the variables x_i by the terms t_i for $i = 1, \ldots, n$. We say that σ is well typed if the variables x_i and the terms t_i have the same types. In the following, we only consider well typed substitutions. The application of the substitution σ on the term t is denoted by $t\sigma$.

We consider a set of predicates $\mathcal{P} = \{P_a | a \in \mathsf{Agent} \cup \{\mathsf{int}\}\}$, where int is a particular predicate representing the knowledge of the attacker.

2.6 A model of corruption

In stating our security properties, we will assume that some keys even if stored on HSMs might be lost, perhaps due to side channel attacks or other out of model events. We will model this as shown in Figure 2 : we assume that an attacker can send arbitrary commands to all HSMs, and additionally, has access to all the keys stored on those HSMs we will refer to as belonging to *dishonest* agents. Other HSMs are referred to as *honest*. This corruption model defines an order relation on the set of keys. To a key k we can associate the set S_k of HSMs the corruption of which implies that of a key. A key k_1 is more secure than k_2 if $S_{k_1} \subseteq S_{k_2}$.

3 Symbolic Security of the API

The model that we present is a transition system inspired by [4]. The state of the system is given by the family $\{\mathcal{S}_b | b \in \mathsf{Agent} \cup \{\mathsf{int}\}\}$. The notations $P_b(t)$ and $t \in \mathcal{S}_b$ are equivalent. The state can evolve by the mean of the rules of the general form :

$$P_1(u_1), \ldots, P_k(u_k) \overset{N_1,\ldots,N_m}{\Longrightarrow} P_{k+1}(v_{k+1}), \ldots, P_l(v_l),$$

where the u_i, v_i are terms, the N_i are variables and the P_i are predicates. We refer to [4] for the definition of an accessible state.

3.1 Security ordering

In order to give the rules of our API, it is necessary to introduce an order relation on the set of triples $(i, S_1, S_2) \in I \times \mathscr{S} \times \mathscr{S}$ (recall that $\mathscr{S}$ is the set of subsets of Agent). Let $(i_1, S_{1,1}, S_{1,2})$ and $(i_2, S_{2,1}, S_{2,2})$ be two elements of $I \times \mathscr{S} \times \mathscr{S}$, we write

$$(i_1, S_{1,1}, S_{1,2}) \prec (i_2, S_{2,1}, S_{2,2}) \text{ if } i_1 < i_2,\ S_{2,1} \subseteq S_{1,1} \text{ and } S_{2,2} \subseteq S_{1,2},$$
$$(i_1, S_{1,1}, S_{1,2}) \preceq (i_2, S_{2,1}, S_{2,2}) \text{ if } i_1 \leq i_2,\ S_{2,1} \subseteq S_{1,1} \text{ and } S_{2,2} \subseteq S_{1,2}.$$

It is clear that $\preceq$ (resp. $\prec$) is an order relation (resp. a strict order relation). This relation plays an important role in the definition of the security policy of our API and the fact that it is strict ensure that we avoid cycles of encryption : for instance we do not want that a term of the form $\{\!\{\{\!\{\ldots\}\!\}_{K_1}\}\!\}_{K_2}\}\!\}_{K_1}$ appears.

This order relation is a natural choice. The security level of the handle is given by the set of HSMs, the corruption of which would imply the corruption of the handle. In the API, we want to guarantee that if a particular set of agents are honest (i.e. have uncorrupted HSMs) then the handle cannot be corrupted. This set was already present in the original Cortier-Steel API, but there only one set was used. In the case of a public key API, the keys are split into a public part (the certificate), whose value is known to everyone but the integrity of which must be guaranteed, and the private part which must be protected in confidentiality and integrity. The security of a key depends on both the two parts, but still it is important to be able to distinguish between these two aspects of security because we want to control the diffusion of the private key even if the integrity of the public part may depend on a long chain of certification.

To makes things clear if (i, S_1, S_2) is a handle level, S_1 should represent a list of agents upon whom the integrity of the public part of the key depends, in other words, if any of these agents are corrupted, then the integrity of the key is lost. On the other hand, S_2 is the set of legitimate users of the private key. If any of these user are corrupted, then the private key is no longer confidential amongst these users. If the handle contains a symmetric key then S_2 has exactly the same meaning as the set S in the Cortier-Steel API.

For asymmetric keys, it may well be the case that S_1 is a rather large set (e.g. tracing a certification chain back to a root certificate) and yet we still want S_2 to be as small as possible (posisbly just the user who generated the key). Finally, it should be remarked that a key k which is wrapped by another asymmetric key k' should inherit from k' the control sets S_1 and S_2 even if k is symmetric.

3.2 The rules of the generic asymmetric API

We are now ready to give the API rules. We illustrate the operation of some rules together with the security ordering in Figure 3.

3.2.1 Rules for symmetric encryption

Symmetric key generation X_k of level i by the agent e for the set of users $S_2 \subseteq$ Agent :

$$i, S_1, S_2 \overset{N, X_k}{\Longrightarrow} P_e(h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \cup \{e\})), \qquad \textbf{(Sym Gen)}$$

with $N \in$ Noncev, $X_k \in$ Keyv, $S_1, S_2 \subseteq$ Agent and $i \in I$.

Symmetric encryption by the agent b with the key X_k of the messages $m_1, \ldots, m_n$:

$$P_b(h_b^a(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2)), P_b(m_1), \ldots, P_b(m_n)$$
$$\Longrightarrow P_b(\{\!|m'_1, \ldots, m'_n|\!\}_{X_k}), \qquad \textbf{(Sym Encrypt)}$$

with $S_1, S_2 \subseteq$ Agent, $b \in S_2$, $N \in$ Noncev, $X_k \in$ Keyv, $i \in I$, $m_j, m'_j \in$ Msg and on the other side $j = 1, \ldots, n$:

- if $m_j = h_b^\alpha(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $X_{k,j} = \mathsf{Keyv} \cup \mathsf{Noncev}$ then
  - if $i \in I_2$, $b \in \mathsf{Agent}$ et $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$ then we have $m'_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$;
  - else $m'_j = \varnothing$.
- else $m'_j = \mathsf{nhdl}(m_j)$.

Symmetric decryption by the agent b with the key X_k :

$$P_b(h_b^\alpha(N, \mathsf{Null}, X_k, \text{symEncDec}, i, S_1, S_2)), P_b(\{\!|m_1, \ldots, m_n|\!\}_{X_k}) \Longrightarrow P_b(m'_1), \ldots, P_b(m'_n), \quad \textbf{(Sym Decrypt)}$$

with $S_1, S_2 \subseteq \mathsf{Agent}$, $b \in S_2$, $N \in \mathsf{Noncev}$, $X_k \in \mathsf{Keyv}$, $i \in I$, $m_j, m'_j \in \mathsf{Msg}$ and on the other side for $j = 1, \ldots, n$:

- if $m_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$, then
  - if $i \in I_2$, $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$ then we set
    $$m'_j = h_b^r(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2});$$
  - else $m'_j = \varnothing$.
- else
  - if $m_j = \mathsf{nhdl}(t_j)$ with $t_j \in \mathsf{Msg}$ then $m'_j = t_j$;
  - sinon $m'_j = \varnothing$.

3.2.2 Rules for asymmetric cryptography

Asymmetric encryption/signature double key generation $(X_k, inv(X_k))$ of level i_2 by the agent e for the user $b \in \mathsf{Agent}$:

$$P_e(h_e^\alpha(N_1, N_2, inv(Y_k), \text{privCertSign}, i_1, S_{e,1}, S_{e,2})), P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_1, S_{e,1}, S_{e,2}))), i_2, S_1, S_2, b \overset{N_3, N_4, X_k}{\Longrightarrow} P_e(h_e^g(N_3, N_4, inv(X_k), \text{privDecSign}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2)), P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubEncVerif}, i_2, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{b, e\} \cup S_2), inv(Y_k))), \quad \textbf{(Asym Gen)}$$

with $N_1, N_2, N_3, N_4, N_{cert} \in \mathsf{Noncev}$, $X_k, Y_k \in \mathsf{Keyv}$, $S_{e,1}, S_{e,2}, S_1, S_2 \subseteq \mathsf{Agent}$, $e \in S_{e,2}$, $b \in \mathsf{Agent}$, $i_1, i_2 \in I_{>0}$, $\alpha \in \{r, g\}$ at the condition that :

- $i_2 < i_1$.

Asymmetric encryption with signature of the agent b for the agent c with the double public key X_k and the signature private key Y_k of the messages $m_1, \ldots, m_n$:

$$P_b(h_b^\alpha(N_1, N_2, inv(Y_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2}), P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, X_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))), P_b(m_1), \ldots, P_b(m_n) \Longrightarrow P_b(\{m'_1, \ldots, m'_n\}_{X_k}), P_b(\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))), \quad \textbf{(Asym SignEncrypt)}$$

with $S_{b,1}, S_{b,2}, S_{c,1}, S_{c,2} \subseteq \mathsf{Agent}$, $i_b, i_c \in I_{>0}$ verifying $(i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2})$, $b \in \mathsf{Agent}$, $c \in S_{c,2}$, $N_1, \ldots, N_5 \in \mathsf{Noncev}$, $X_k, Y_k \in \mathsf{Keyv}$, $m_j, m'_j \in \mathsf{Msg}$ and on the other side for $j = 1, \ldots, n$:

- if $m_j = h_b^{\alpha}(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$ then :
  - if $i_b, i_c \in I_2$ and $(i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2})(\preceq (i_c, S_{c,1}, S_{c,2}))$ then
$$m'_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2};$$
  - else $m'_j = \varnothing$.
- else $m'_j = \mathsf{nhdl}(m_j)$.

Asymmetric decryption with signature verification by the agent b with the decryption private key $inv(X_k)$ and the signature verification public key Y_k :

$$\begin{aligned} &P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),\\ &\qquad P_b(h_b^{\alpha}(N_4, N_5, inv(X_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2})),\\ &\qquad P_b(\{m_1, \ldots, m_n\}_{X_k}), P_b(\Sigma(\{m_1, \ldots, m_n\}_{X_k}, inv(Y_k)))\\ \Longrightarrow &P_b(m'_1), \ldots, P_b(m'_n), \end{aligned} \quad \textbf{(Asym VerifDecrypt)}$$

with $S_{b,1}, S_{b,2}, S_{c,1}, S_{c,2} \subseteq \mathsf{Agent}$, $i_b, i_c \in I_{>0}$, verifying $(i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2})$, $b \in \mathsf{Agent}$, $N_1, \ldots, N_5 \in \mathsf{Noncev}$, $X_k, Y_k \in \mathsf{Keyv}$, $m_j, m'_j \in \mathsf{Msg}$ and on the other side for $j = 1, \ldots, n$:

- if $m_j = N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}$ then
  - if $i_b, i_c \in I_2$, $b \in \mathsf{Agent}$ and $(i_j, S_{j,2}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2})(\preceq (i_c, S_{c,1}, S_{c,2}))$ then
$$m'_j = h_b^r(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2});$$
  - else $m'_j = \varnothing$.
- if $m_j = \mathsf{nhdl}(t_j)$ for $t_j \in \mathsf{Msg}$ then $m'_j = t_j$.

3.2.3 Certificate management rules

Certification key generation $(X_k, inv(X_k))$ of level i_b by the agent e for the agent b :

$$\begin{aligned} &P_e(h_e^{\alpha}(N_1, N_2, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2})),\\ &\qquad P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))), i_b, S_1, S_2 \overset{N_3, N_4, X_k}{\Longrightarrow}\\ &\qquad P_e(h_e^g(N_3, N_4, inv(X_k), \text{privCertSign}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2)),\\ &P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubCertVerif}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2), inv(Y_k))), \end{aligned}$$

(Cert Gen)

with $N_1, N_2, N_3, N_4, N_{cert} \in \mathsf{Noncev}$, $X_k, Y_k \in \mathsf{Keyv}$, $S_{e,1}, S_{e,2}, S_1, S_2 \subseteq \mathsf{Agent}$, $b \in \mathsf{Agent}$, $e \in S_{e,2}$, $i_e \in I_2$ and at the condition that

- $i_b \leq i_e$.

Verification of a certificate by the agent b : Pour $\Theta \in \{\text{EncVerif}, \text{CertVerif}\}$,

$$P_b(\Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k))),$$
$$P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2})))\Longrightarrow$$
$$P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))), \qquad \textbf{(Cert Verif)}$$

with $b \in \mathsf{Agent}$, $N_1, \ldots, N_5 \in \mathsf{Noncev}$, $X_k, Y_k, inv(Y_k) \in \mathsf{Keyv}$, $S_{c,1}, S_{c,2}, S_{e,1}, S_{e,2} \subset \mathsf{Agent}$, $i_c, i_e \in I_{>0}$ at the condition that :

- $(i_c, S_{c,1}, \varnothing) \prec (i_e, S_{e,1} \cup S_{e,2}, \varnothing)$.

3.2.4 Example

In Figure 3 we show the 'before' and 'after' states for three agents using the API in a typical configuration. In the 'before' state, there are no shared secrets. Alice and Bob both have accepted a copy of the CA's public key certificate and placed it under an integrity handle (identifiers 5 and 8 respectively) and they have generated their own public-private keypairs. The CA has accepted public key certificates for each of these pairs (identifiers 3 and 4). Here we are using integers to label key levels, arbitrarily assigning the long term keys the level 3. Public keys are level 0.

To establish a shared secret, Alice and Bob first need to accept each others public key certificates. This can be done by requesting them from the CA. The CA uses the AsymEncryptSign command to sign the (public) message containging the certificate. Now Alice and Bob can use the certificate verification command to accept the certificates, generating handles 11 and 12.

Now either Alice can generate a symmetric key (handle 13) and send it to Bob using AsymEncryptSign. Bob will use AsymDecryptVerify and accept the key (handle 14). Now Alice and Bob can exchange messages using the new symmetric key. Note that the new symmetric key is confidential between Alice and Bob, hence has a confidentiality control set S_2 containing only these identifiers, but for integrity it has inherited the dependence on the CA, hence S_1 contains the set of agents CA, Alice and Bob.

4 Security of the API in the symbolic model

4.1 Model of security

In this section, we describe the capacity of the attacker in the spirit of the Dolev-Yao [1] model.

4.1.1 Computation of new terms

We denote by INTRUDER the set of rules which allow the attacker to build new terms from the ones that it has already.

Creation/destruction of pairs Let $m_1, m_2 \in \mathsf{Msg}$, we have:

- $P_{\mathsf{int}}(m_1), P_{\mathsf{int}}(m_2) \Rightarrow P_{\mathsf{int}}(< m_1, m_2 >)$
- $P_{\mathsf{int}}(< m_1, m_2 >) \Rightarrow P_{\mathsf{int}}(m_1), P_{\mathsf{int}}(m_2)$

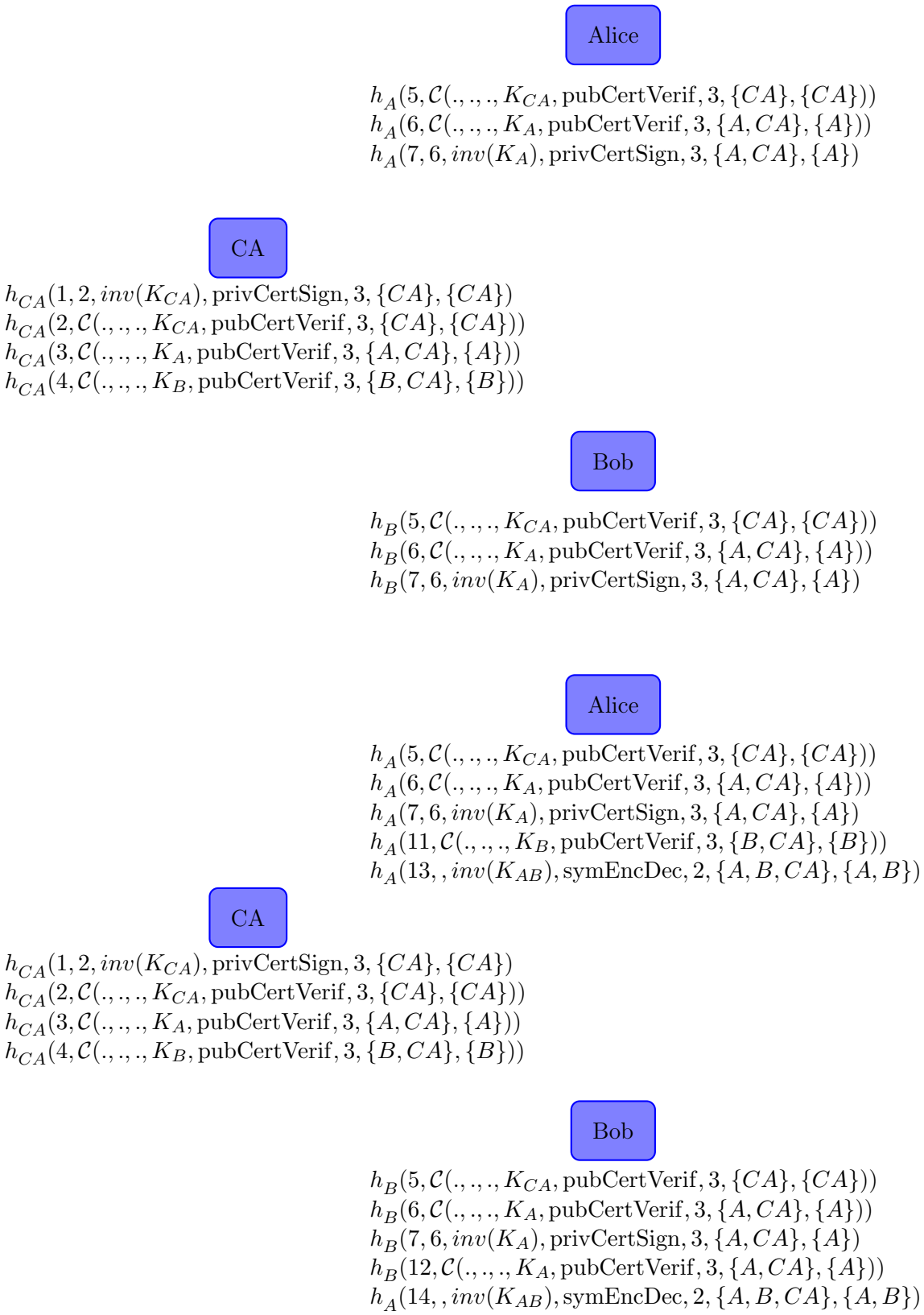


Figure 3: Operation of the API. See 3.2.4 for narration.

Symmetric encryption/decryption let $X_k \in \mathsf{Keyv}$, $m_1, \ldots, m_n \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n) \Rightarrow P_{\mathsf{int}}(\{\!|m_1, \ldots, m_n|\!\}_{X_k})$
- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(\{\!|m_1, \ldots, m_n|\!\}_{X_k}) \Rightarrow P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n)$

Asymmetric encryption/decryption Let $X_k \in \mathsf{Keyv}$, $m_1, \ldots, m_n \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n) \Rightarrow P_{\mathsf{int}}(\{m_1, \ldots, m_n\}_{X_k})$
- $P_{\mathsf{int}}(inv(X_k)), P_{\mathsf{int}}(\{m_1, \ldots, m_n\}_{X_k}) \Rightarrow P_{\mathsf{int}}(m_1), \ldots, P_{\mathsf{int}}(m_n)$

Signature Let $X_k \in \mathsf{Keyv}$, $m \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(X_k), P_{\mathsf{int}}(m) \Rightarrow P_{\mathsf{int}}(\Sigma(m, X_k))$
- $P_{\mathsf{int}}(\Sigma(m, X_k)) \Rightarrow P_{\mathsf{int}}(m)$

Creation/destruction of container of regular messages Let $m \in \mathsf{Msg}$, we have :

- $P_{\mathsf{int}}(m) \Rightarrow P_{\mathsf{int}}(\mathsf{nhdl}(m))$
- $P_{\mathsf{int}}(\mathsf{nhdl}(m)) \Rightarrow P_{\mathsf{int}}(m)$

The transitive reflexive closure of the preceding rules can be interpreted as the set of terms that an attacker can deduce from its knowledge at a certain state. In the following, we say that m is deducible from a set of terms T, that we denote by $T \vdash m$, if starting from the state $\mathcal{S}$ such that $\mathcal{S}_{\mathsf{int}} = T$ and for all $a \in \mathsf{Agent}$, $\mathcal{S}_a = \emptyset$, there exists a state $\mathcal{S}'$ such that $\mathcal{S} \Longrightarrow^*_{\mathsf{INTRUDER}} \mathcal{S}'$ and $m \in \mathcal{S}'_{\mathsf{int}}$. In the following, we will do the following abuse of notation : if t is a term and $\mathcal{S}$ is a state, we write $t \in \mathcal{S}$ (resp. $\mathcal{S} \vdash t$) if $t \in \cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_b$ (resp. if $\cup_{b \in \mathsf{Agent} \cup \{\mathsf{int}\}} \mathcal{S}_b \vdash t$).

4.1.2 Control of the network and corruption

Control of the network Let a be an agent m a message.

- $P_a(m) \Rightarrow P_{\mathsf{int}}(m)$
- $P_{\mathsf{int}}(m) \Rightarrow P_a(m)$

In the following, we suppose given a set H a honest agents.

Corruption of a HSM Let a be a dishonest agent (i.e. $a \notin H$) and

$$h^{\alpha}_a(N_1, N_2, m, T, i, S_1, S_2),$$

a handle then

$$P_a(h^{\alpha}_a(N_1, N_2, m, T, i, S_1, S_2)) \Rightarrow P_{\mathsf{int}}(m).$$

4.2 Initial states

We are going to consider the transition system defined by the set of rules of the API from a starting state. In order to do so, it is necessary that this initial state verify certain properties. Indeed, it is clear that it is not possible to prove a non trivial security property if we don't have reasonable assumptions on the initial state. On the other side, it seems reasonable to impose realistic conditions for the initialisation of the HSM which can be done in a controlled environment. We would like also to put to the knowledge of the attacker certain public informations like the set of key levels, the set of agents. These considerations motivate the following definition.

Definition 1. A state $\mathcal{S}_0$ is said to be *initial* if it satisfies the following hypotheses :

1. the set of terms known by the agents and the intruder are atomic : for all $a \in \mathsf{Agent} \cup \{\mathsf{int}\}$, $\mathcal{S}_a \subseteq \mathsf{Handle} \cup \mathsf{Key} \cup \mathsf{Nonce} \cup \mathsf{Agent} \cup \mathcal{T} \cup I \cup \mathscr{S}$ and moreover $\mathcal{T} \cup I \cup \mathscr{S} \subseteq \mathcal{S}_{\mathsf{int}}$.
2. all terms stored under honest handles are secret : for $a \in \mathsf{Agent}$, if $h_a^\alpha(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ then for $b \in \mathsf{Agent} \cup \{\mathsf{int}\}$, $m \notin \mathcal{S}_b$.
3. all key handle known by an agent point to an atomic element : for $a \in \mathsf{Agent}$, if $h_a^\alpha(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ then $m \in \mathsf{Key} \cup \mathsf{Nonce}$.
4. the owner of a key handle is in the control set. More precisely, we impose that for all $a \in \mathsf{Agent}$, if $h_a^\alpha(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ alors $a \in S_2$.
5. the public key certificates under handle owned by an agent corresponds to a private key store by another (or the same) agent. For all $b \in \mathsf{Agent}$, if $h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S_1, S_2)) \in \mathcal{S}_b$, then there exists $a \in S_2$ so that
$$h_a^\alpha(N_2, N_1, inv(X_k), priv\Theta', i, S_1, S_2) \in \mathcal{S}_a,$$
with $(\Theta, \Theta') \in \{(EncVerif, DecSign), (CertVerif, CertSign)\}$.
6. the key handles form a *coherent* set: for all $a, a' \in \mathsf{Agent}$, $h_a^\alpha(N_1, N_2, m, T, i, S_1, S_2) \in \mathcal{S}_a$ and $h_{a'}^{\alpha'}(N_1', N_2', m, T', i', S_1', S_2') \in \mathcal{S}_{a'}$ then $N_1 = N_1', T = T', i = i', N_2 = N_2', S_1 = S_1'$ and $S_2 = S_2'$.

We can now define the set of states for which we can prove a security property.

Definition 2. We say that a state is *accessible* from an initial state $\mathcal{S}_0$ if it is attainable by applying a finite number of times the rules of the set API, INTRUDER and CONTROL to $\mathcal{S}_0$.

4.3 Security criterion sketch of proof

The security of the API can be expressed in the following way: given a state $\mathcal{S}$, the secret data of honest agent should not be known to the intruder. But we would like also to be sure that the secret data is only used by rightful agents. As the secret data of honest users are the messages $m \in \mathsf{Msg}$ for which there exists a handle of the form $h_a^\alpha(.,.,m,.,.,S_1,S_2)$ with $a \in H$ and $S_1, S_2 \subseteq H$, and as the set of legitimate users of m is S_2 the property that we want to prove is reflected by the formula:

$$\forall a \in H, \forall m \in \mathsf{Msg}, \forall i \in I_{>0}, \forall \alpha \in \{r, g\}, \forall S_1, S_2 \subseteq H,\ \mathcal{S} \vdash h_a^\alpha(.,.,m,.,i,S_1,S_2) \Rightarrow \mathcal{S} \nvdash m \text{ and } a \in S_2 \quad \textbf{(Sec)}$$

We can now give the principal result of this note which say that our API verify the property (**Sec**) if it is correctly initialised.

Theorem 1 (Confidentiality of handles)**.** Let $\mathcal{S}_0$ be an initial state. Let $\mathcal{S}$ be an accessible state from an initial state $\mathcal{S}_0$, which means that $\mathcal{S}_0 \Rightarrow^*_{\mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}} \mathcal{S}$, then $\mathcal{S}$ satisfy the property (**Sec**).

Proof. We present a scheme of proof which will be detailed in the Section 5. We first begin to consider a more powerful attacker which has access to all the value stores in the compromised HSM as well as to all the messages m associated to handles of the form $h_a^\alpha(.,.,m,.,.,S_1,S_2)$ where $S_1, S_2 \subsetneq H$ even if a is honest. On the other side, we gather the set of all the terms known by all the agent in one big set since the intruder has the ability to pass a term from the knowledge of an agent to the knowledge of another via the CONTROL rules.

More precisely, we write $\mathcal{S} \vdash^* t$ when $\cup_{b\in \mathsf{Agent}\cup\{\mathsf{int}\}}\mathcal{S}_b \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in \mathcal{S}, S_1 \subsetneq H$ ou $S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., ., .) \in \mathcal{S}, a \notin H\} \cup \{m | h_a(., m) \in \mathcal{S}\} \vdash t$.

We then consider a stronger version of the perperty (**Sec**):

$$\forall a \in H, \forall m \in \mathsf{Msg}, \forall i \in I_{>0}, \forall \alpha \in \{r, g\}, \forall S_1, S_2 \subseteq H,\ \mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S_1, S_2) \Rightarrow \mathcal{S} \nvdash^* m, \quad a \in S_2 \text{ and } m \in \mathsf{Key} \cup \mathsf{Nonce}. \quad (\mathbf{Sec}^*)$$

Intuitively, the property (**Sec***) means that the values stored in the handles of honest agents are always of type key or nonce and are not deducible even with the extended deduction rule $\vdash^*$. It is clear that in order to prove the theorem, it is enough to prove the same statement with the stronger version of the property (**Sec**).

In the Section 5, we prove by induction that the property (**Sec***) together with the following properties are invariant when applying the rules $\mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$. What we mean is that if

$\mathcal{S}'$ is so that $\mathcal{S} \Rightarrow_R \mathcal{S}'$ for $R \in \mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$ and if $\mathcal{S}$ satisfies the properties (**Sec***) and the following ones, then it is also the case for $\mathcal{S}'$.

$$\forall a \in H, \forall N_1, N_2, N_3 \in \mathsf{Nonce}, \forall i \in I_{>0}, \forall S_1, S_2 \subseteq H \text{ with}$$
$$\mathcal{S} \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S_1, S_2)) \Rightarrow \exists b \in S_2 \text{ such that } \mathcal{S} \vdash^* h_b^\alpha(N_2, N_1, inv(X_k), priv\Theta', i, S_1, S_2). \quad (\mathbf{Cert})$$

où $(\Theta, \Theta') \in \{(\mathrm{EncVerif}, \mathrm{DecSign}), (\mathrm{CertVerif}, \mathrm{CertSign})\}$.

Intuitively, the property (**Cert**) means that if a certificate is under an integrity handle with control sets S_1, S_2 containing only honest users, then there exists a private key handle associated to this certificate the attributes of which are coherent with that of the certificate.

$$\forall u, k \in \mathsf{Msg}, \mathcal{S} \vdash^* \{\!|u|\!\}_k \Rightarrow \mathcal{S} \vdash^* k \textbf{ or}$$
$$\exists i \in I_{>0}, \exists S_1, S_2 \subset H\ \exists a \in S_2 \text{ such that } \mathcal{S} \vdash^* h_a(., ., k, ., i, S_1, S_2)$$
$$\exists u'_1, \ldots, u'_p \in \mathsf{Msg} \text{ such that } u = u'_1, \ldots, u'_p \text{ with } \forall j = 1 \ldots p,$$
$$\textbf{either } \exists m_j \in \mathsf{Key} \cup \mathsf{Nonce}, \exists T_j \in T, \exists (i_j, S_{j,1}, S_{j,2}) \in (I \times \mathscr{S} \times \mathscr{S}), \exists (N_{j,1}, N_{j,2}) \in \mathsf{Nonce}^2 \text{ such that}$$
$$u'_j = N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2},\ (i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2) \text{ and}$$
$$\mathcal{S} \vdash^* h_a(N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2})$$
$$\textbf{or } \exists m_j \in \mathsf{Msg} \text{ such that } m'_j = \mathsf{nhdl}(m_j) \text{ and } \mathcal{S} \vdash^* m_j. \quad (\mathbf{SymEnc})$$

Intuitively, the property (**SymEnc**) means that every message encrypted with a symmetric key either can be built by the attacker, or corresponds to a valid output of the symmetric encryption rule.

$$\forall u, k \in \mathsf{Msg}, \mathcal{S} \vdash^* \Sigma(u,k) \Rightarrow \mathcal{S} \vdash^* k \textbf{ or}$$
$$\exists b \in \mathsf{Agent}, \exists e \in \mathsf{Agent}, \exists i_1, i_2 \in I, \exists N_2, N_3, N_4 \in \mathsf{Nonce}, \exists S_1', S_2', S_1, S_2 \subset \mathsf{Agent}, \exists X_k \in \mathsf{Keyv} \text{ such that}$$
$$u = \mathcal{C}(N_4, N_3, N_2, X_k, \mathsf{pub}\Theta, i_2, S_1 \cup \{e\}, S_2 \cup \{b, e\}) \text{ and}$$
$$\mathcal{S} \vdash^* h_e^{\cdot}(.,.,k, \text{privCertSign}, i_1, S_1', S_2') \text{ with } S_1' \cup S_2' \subset S_1,\ e \in S_2' \subset \mathsf{Agent}, i_2 < i_1$$
$$\textbf{or}$$
$$\exists S_{b,1}, S_{b,2} \subset \mathsf{Agent}, \exists b \in S_{b,2}, \exists i_b \in I \text{ such that } \mathcal{S} \vdash^* h_b(.,.,k, \text{privDecSign}, i_b, S_{b,1}, S_{b,2})$$
$$\exists S_{c,1}, S_{c,2} \subset \mathsf{Agent}, \exists i_c \in I,\ \exists K \in \mathsf{Key} \text{ such that}$$
$$\mathcal{S} \vdash^* h_b(., \mathcal{C}(.,.,.,K, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))$$
$$\exists u_1', \ldots, u_p' \in \mathsf{Msg} \text{ such that } u = u_1', \ldots, u_p' \text{ with } \forall j = 1 \ldots p,$$
$$\textbf{either } \exists m_j \in \mathsf{Key} \cup \mathsf{Nonce}, \exists T_j \in T, \exists (i_j, S_{j,1}, S_{j,2}) \in (I \times \mathscr{S} \times \mathscr{S}), \exists (N_{j,1}, N_{j,2}) \in \mathsf{Nonce}^2 \text{ such that}$$
$$u_j' = N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2},\ (i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2}) \text{ and}$$
$$\mathcal{S} \vdash^* h_b^{\cdot}(N_{j,1}, N_{j,2}, m_j, T_j, i_j, S_{j,1}, S_{j,2})$$
$$\textbf{or } \exists m_j \in \mathsf{Msg} \text{ such that } m_j' = \mathsf{nhdl}(m_j) \text{ and } \mathcal{S} \vdash^* m_j$$
$$\textbf{or}$$
$$\exists b \in H, \exists S_1, S_2 \subset H \text{ such that } \mathcal{S} \vdash^* h_b^{\cdot}(., k, ., ., ., S_1, S_2),\ h_b(., u') \text{ with } u = \mathsf{nhdl}(u') \qquad \textbf{(Sign)}$$

for $\Theta \in \{\text{EncVerif}, \text{CertSign}\}$.

Intuitively, the property (**Sign**) means that every signature sent over the network can either be built by the attacker, or corresponds to a valid output of encryption/signature or certificate asymmetric key generation rule, or of asymmetric encryption rule.

To finish the sketch proof, we remark moreover that from its definition, an initial state satisfies the properties (**Sec***), (**Cert**), (**SymEnc**), (**Sign**) (see the section 5 for more details). □

5 Detailed proof

In this section, we provide a detailed proof of Theorem 1. We want to show that the properties (**Sec***), (**Cert**), (**SymEnc**), (**Sign**) are left invariant by application of the rules $\mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$. We remark that all the transitions coming from the rules API create fresh handles, symmetric and asymmetric encrypted messages, or signatures. The following technical lemma is a useful tool which will be repeatedly used during the proof, since it says that from handles, encrypted messages the decryption key of which is unknown to him and from signatures, the attacker can not create new terms which are very useful for its purpose.

Lemma 1. let $k_1, k_2 \in \mathsf{Key}$ and let $t_1, t_2 \in \mathsf{Msg}$. We denote by f the symmetric or asymmetric encryption function. Let S and S' be sets of terms such that

$$S' \subseteq S \cup \{f(t_1, k_1)\} \cup \ \{\Sigma(t_2, k_2)\} \cup \mathsf{Hdls} \cup \mathsf{NK}$$

with NK a set of elements of $\mathsf{Key} \cup \mathsf{Nonce} \cup \mathsf{Agent} \cup \mathcal{T} \cup I \cup \mathscr{S}$ which do not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$ and $\mathsf{Hdls} \subseteq \mathsf{Handle}$ a set of handles which are either

- of the form $h_a^{\alpha}(.,.,k,.,.,S_1,S_2)$ with $S_1 \cup S_2 \cup \{a\} \subseteq H$, or $S_1 \cup S_2 \cup \{a\} \nsubseteq H$, and then we suppose that $S \vdash^* k$ or that $k \in \mathsf{NK}$;
- or of the form $h_a(., N)$ with $a \in \mathsf{Agent}$ and $S \vdash^* N$ or $\mathsf{NK} \vdash N$.

We suppose moreover that $S \cup \{f(t_1, k_1)\} \cup \mathsf{NK} \vdash^* t_2$ and that k_1', with $k_1' = k_1$ if k_1 is a symmetric key (resp. $k_1' = inv(k_1)$ if k_1 is an asymmetric key) and k_2 are not deducible, that is to say that $S \nvdash^* k_1'$, $S \nvdash^* k_2$, $k_1', k_2 \notin \mathsf{NK}$. Let $u \in \mathsf{Agent} \cup \mathsf{Nonce} \cup \mathsf{Key} \cup \mathsf{Handle}$ be an atomic message. We have :

$$S' \vdash^* u \text{ if and only if } S \vdash^* u \text{ or } u \in \mathsf{Hdls} \text{ or } u \in \mathsf{NK}. \tag{1}$$

Moreover, let $v \in \mathsf{Msg}$ and $w \in \mathsf{Key}$, then we have

$$S' \vdash^* \{\!|v|\!\}_w \text{ and } S' \nvdash^* w \text{ if and only if } S \vdash^* \{\!|v|\!\}_w \text{ or} \{\!|v|\!\}_w = f(t_1, k_1), \tag{2}$$

$$S' \vdash^* \Sigma(v, w) \text{ and } S' \nvdash^* w \text{ if and only if } S \vdash^* \Sigma(v, w) \text{ or} f(v, w) = \Sigma(t_2, k_2). \tag{3}$$

Remark 1. In the statement, when we say that an element of NK does not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$ this means in particular that if $inv(k) \in \mathsf{NK}$ then k does not appear in $S \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\}$.

Proof. We let $S_\infty = \{u | S \vdash^* u\}$, $S_0' = S' \cup S_\infty \subseteq S_\infty \cup \{f(t_1, k_1)\} \cup \{\Sigma(t_2, k_2)\} \cup \mathsf{Hdls} \cup \mathsf{NK}$ and we consider the sets $(S_i')_{i \in \mathbb{N}}$ defined inductively by:

$$S_{i+1}' = S_i' \cup \{u | S_i' \cup \Delta_i \Rightarrow_R u, R \in \mathsf{INTRUDER}\},$$

where $\Delta_i = \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in S_i', S_1 \subsetneq H \text{ or } S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., ., .) \in S_i', a \notin H\} \cup \{m | h_a(., m) \in S_i'\}$. On the other side, we let $F_i' = S_i' - S_\infty$. We show by induction that

$$u \in (\mathsf{Agent} \cup \mathsf{Nonce} \cup \mathsf{Key} \cup \mathsf{Handle}) \cap F_i' \Rightarrow u \in \mathsf{Hdls} \cup \mathsf{NK}, \tag{4}$$

$$\forall v \in \mathsf{Msg}, \forall w \in \mathsf{Key}, \Sigma(v, w) \in S_i' \text{ and } w \notin S_i' \Rightarrow \Sigma(v, w) \in S_\infty \text{ or } \Sigma(v, w) = \Sigma(t_2, k_2), \tag{5}$$

$$\forall v \in \mathsf{Msg}, \forall w \in \mathsf{Key}, \{\!|v|\!\}_w \in S_i' \text{ and } w \notin S_i' \Rightarrow \{\!|v|\!\}_w \in S_\infty \text{ or } \{\!|v|\!\}_w = f(t_1, k_1). \tag{6}$$

At the same time, we prove that S_i' is included in the union of

$$A = S_\infty \cup f(t_1, k_1) \cup \Sigma(t_2, k_2) \cup \mathsf{Hdls} \cup \mathsf{NK}, \tag{7}$$

$$B_i = < m_1, m_2 > \text{ with } m_1, m_2 \in S_{i-1}', \tag{8}$$

$$C_i = \{\!|m_1, \ldots, m_n|\!\}_K \text{ with } m_1, \ldots, m_n, K \in S_{i-1}', \tag{9}$$

$$D_i = \{m_1, \ldots, m_n\}_K \text{ with } m_1, \ldots, m_n, K \in S_{i-1}', \tag{10}$$

$$E_i = \Sigma(m_1, K) \text{ with } m_1, K \in S_{i-1}', \tag{11}$$

$$F_i = \mathsf{nhdl}(m) \text{ with } m \in S_{i-1}'. \tag{12}$$

It is clear that the induction hypothesis is verified for S_0' (by setting $S_{-1}' = \emptyset$). Suppose that the hypothesis are true for S_i', we prove the hypothesis for S_{i+1}'.

We begin by remarking that because of the hypothesis made on the handles of Hdls, we have $\Delta_i \subset S_\infty \cup \mathsf{NK}$. Thus, we have $S_{i+1}' = S_i' \cup \{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\}$. For all $i \in \mathbb{N}$, set $H_i = A \cup B_i \cup C_i \cup D_i \cup E_i \cup F_i$. We first prove that $S_{i+1}' \subseteq H_{i+1}$. For this, we remark that from the induction hypothesis, $S_i' = H_i \subseteq H_{i+1}$ and by definition $S_{i+1}' = S_i' \cup \{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\}$. Thus, it suffices to check that $\{u | S_i' \Rightarrow_R u, R \in \mathsf{INTRUDER}\} \subseteq H_{i+1}$ by running through all the rules of $\mathsf{INTRUDER}$ in order to prove that $S_{i+1}' \subseteq H_{i+1}$:

- pair creation : this corresponds to the set B_{i+1}.
- pair destruction : as $S_i' = H_i$, every pair of S_i' is either given by the set B_i or is in S_∞. In all cases, an application of the pair destruction rule is going to output an element of $S_{i-1}' \subseteq S_i' \subseteq H_{i+1}$.
- symmetric encryption : this corresponds to the set C_{i+1}.

- symmetric decryption : because of (4) and $S \not\vdash^* k_1'$, k_1' (with the notation of the statement of the lemma) is not in S_i'. By applying the induction hypothesis for $S_i' = H_i$, the decryption rule can only apply to a term of C_i or a term of S_∞. For the first case, this gives elements of $S_{i-1}' \subseteq S_i' \subseteq H_{i+1}$. In the second case, the term is necessarily of the form $\{\!|\, m \,|\!\}_k$ with $k \in S_\infty$ by applying (4) and the fact that the elements of NK do not appear in S. Thus we have $m \in S_\infty \subseteq H_{i+1}$.

- asymmetric encryption : this corresponds to the set D_{i+1}.

- asymmetric decryption : because of (4), k_1' is not in S_i'. Thus, by applying the induction hypothesis $S_i' \subseteq H_i$, the decryption rule can only apply to a term of the set D_i or to a term of S_∞ . In the first case, we obtain elements of $S_{i-1}' \subseteq S_i' \subseteq H_{i+1}$. In the second case, the term is necessarily of the form $\{m\}_k$ with $inv(k) \in S_\infty$ by applying (4) and because of the fact that the elements of NK do not appear in S. Thus we have $m \in S_\infty \subseteq H_{i+1}$.

- signature : this corresponds to the set E_{i+1}.

- signature message : by application of the induction hypothesis $S_i' \subseteq H_i$, the message signature rule can only apply to a term of S_∞, to the set E_i or to $\Sigma(t_2, k_2)$. In the first case, this gives a term of S_∞, in the second case, we obtain a term of $S_{i-1}' \subseteq S_i' \subseteq H_{i+1}$ and in the last case, this gives t_2 which is by hypothesis deducible from $S_\infty \cup \mathsf{NK}$.

We can now prove the other hypothesis :

- hypothesis (4) : as $S_{i+1} \subseteq H_{i+1}$, we know that S_{i+1}' is included in the union of $S_\infty \cup f(t_1, k_1) \cup \Sigma(t_2, k_2) \cup \mathsf{Hdls} \cup \mathsf{NK} \subseteq S_i'$ and of non atomic terms which allows to conclude by applying (4).

- hypothesis (6) : if $\{\!|v|\!\}_w \in S_{i+1}'$ and $w \notin S_{i+1}'$ then $\{\!|v|\!\}_w$ is not in the set C_{i+1} and thus $\{\!|v|\!\}_w$ is in the set A which allows to conclude.

- hypothesis (5) : ditto the preceding proof.

In order to finish the proof, we just have to remark that by definition $\cup_{i\in\mathbb{N}} S_i' = \{u | S' \vdash^* u\}$. □

First, we use Lemma 1 in order to check that every initial state (see Definition 1) satisfies the properties (**Sec***), (**Cert**), (**SymEnc**), (**Sign**). Let $\mathcal{S}_0$ be an initial state. Let

$$A = \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in \mathcal{S}_0, S_1 \subsetneq H \text{ or } S_2 \subsetneq H, a \in \mathsf{Agent}\} \cup \{m, N_1, N_2 | h_a^\alpha(N_1, N_2, m, ., ., ., .) \in \mathcal{S}_0, a \notin H\} \cup \{m | h_a(., m) \in \mathcal{S}_0\}.$$

Now writing $\mathcal{S}_{0,b}$ for b's knowledge in $\mathcal{S}_0$, we let

$$\mathsf{NK} = A \cup_{b\in\mathsf{Agent}\cup\{\mathsf{int}\}} (\mathcal{S}_{0,b} - \mathsf{Handle}),$$

$$\mathsf{Hdls} = \cup_{b\in\mathsf{Agent}\cup\{\mathsf{int}\}} \mathcal{S}_{0,b} \cap \mathsf{Handle}.$$

Let $S' = \mathsf{NK} \cup \mathsf{Hdls}$. It is clear from the definition of $\vdash^*$ that $\{u | \mathcal{S}_0 \vdash^* u\} = \{u | S' \vdash^* u\}$. Let $h \in \mathsf{Handle}$ be so that $S' \vdash^* h$, from the Lemma 1 conclusion (1) (since $S = \emptyset$, we have $h \in \mathsf{Hdls}$. Suppose now that $h = h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2)$. The Lemma 1 conclusion (1) tells us that if $S' \vdash^* m$ then $m \in \mathsf{NK}$ which is only possible if $\{a\} \cup S_1 \cup S_1 \not\subseteq H$ from the Definition 1. We deduce that for all $h_a^\alpha(N_1, N_2, m, ., ., S_1, S_2) \in \mathsf{Handle}$ so that $\{a\} \cup S_1 \cup S_2 \subseteq H$, $S' \not\vdash^* m$ and $m \in \mathsf{Keyv} \cup \mathsf{Noncev}$ which is exactly property (**Sec***). By applying Lemma 1 conclusion (2), we obtain that if $S' \vdash^* \{\!|v|\!\}_w$ then necessarily $S' \vdash^* w$ which proves (**SymEnc**). Ditto for the signature, Lemma 1 conclusion (3) tells us that if $S' \vdash^* \Sigma(u, k)$ then $S' \vdash^* k$ which proves property (**Sign**). To finish, (**Cert**) is an immediate

consequence of the definition of an initial state and of the fact that every deducible handle of S' is in Hdls from Lemma 1 conclusion (1).

We are ready to prove Theorem 1. Let $\mathcal{S}$ be a state satisfying the properties (**Sec***), (**Cert**), (**SymEnc**), (**Sign**), and let $\mathcal{S}'$ be such that $\mathcal{S} \Rightarrow_R \mathcal{S}'$ for $R \in \mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$. We want to prove that $\mathcal{S}'$ also satisfies the properties. We see that it suffices to check this for each $R \in \mathsf{API} \cup \mathsf{INTRUDER} \cup \mathsf{CONTROL}$.

It is immediate to see that the properties (**Sec***), (**Cert**), (**SymEnc**), (**Sign**) are left invariant by application of the rules of $\mathsf{INTRUDER}$ and $\mathsf{CONTROL}$. Indeed, these properties are about terms deducible from $\mathcal{S}$ but by definition of the deducibility relation, we have $\{u|\mathcal{S} \vdash^* u\} = \{u|\mathcal{S}' \vdash^* u\}$ with $\mathcal{S} \Rightarrow_R \mathcal{S}'$ and $R \in \mathsf{INTRUDER}$. Ditto from the definition of the extended deducibility relation, we have $\{u|\mathcal{S} \vdash^* u\} = \{u|\mathcal{S}' \vdash^* u\}$ with $\mathcal{S} \Rightarrow_R \mathcal{S}'$ and $R \in \mathsf{CONTROL}$.

It only remains to see that the properties are left invariant if $R \in \mathsf{API}$ which is done by checking each rule. In the following, we suppose that $\mathcal{S}$ satisfies the properties and we define $\mathcal{S}'$ as $\mathcal{S} \Rightarrow_R \mathcal{S}'$ with $R \in \mathsf{API}$.

The rule (Sym Gen) : Reminder of the rule (with the notations of the Section 3.2) :

$$i, S_1, S_2 \quad \stackrel{N,X_k}{\Longrightarrow} \quad P_e(h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \quad \cup \quad \{e\}))$$

We apply Lemma 1 to $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \cup \{e\})\} \cup \mathsf{NK}$ with $\mathsf{NK} = \{X_k\}$ if $\{e\} \cup S_1 \cup S_2 \not\subseteq H$ and $\mathsf{NK} = \emptyset$ on the contrary.

- invariance of (**Sec***) : let $h_a^\alpha(.,.,m,.,i,S'_1,S'_2) \in \mathsf{Handle}$ with $a \in H$, $S'_1, S'_2 \subseteq H$ be such that $\mathcal{S}' \vdash^* h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$ then conclusion (1) of Lemma 1 tells us that either
  - $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$,
  - or $h_a^\alpha(.,.,m,.,i,S'_1,S'_2) = h_e^g(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2 \cup \{e\})$.

  In the first case, the induction hypothesis allows us to conclude. In the second case, Lemma 1 conclusion (1) with $u = X_k$ implies that $\mathcal{S}' \nvdash^* X_k$ and on the other side it is clear that $X_k \in \mathsf{Keyv}$ and $e \in S_2 \cup \{e\}$.

- invariance of (**Cert**) : from the conclusion (1) of Lemma 1, if

  $$\mathcal{S}' \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$$

  then $\mathcal{S} \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$ and the induction hypothesis allows us to conclude.

- invariance of (**SymEnc**) : from the conclusion (2) of Lemma 1, if $\mathcal{S}' \vdash^* \{\!|u|\!\}_k$ and $\mathcal{S}' \nvdash^* k$ then $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and the induction hypothesis concludes.

- invariance of (**Sign**) : from conclusion (3) of Lemma 1, $\mathcal{S}' \vdash^* \Sigma(u, inv(k))$ and $\mathcal{S}' \nvdash^* inv(k)$ implies that $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and the induction hypothesis conclude.

The rule (Sym Encrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$P_b(h_b^\alpha(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2)), P_b(m_1), \ldots, P_b(m_n)$$
$$\Longrightarrow P_b(\{\!|m'_1, \ldots, m'_n|\!\}_{X_k})$$

If $\mathcal{S} \vdash^* X_k$, then from hypothesis (**Sec***), it means that either $b \notin H$, or $S_1, S_2 \not\subseteq H$. In this case, if m_j is a handle of the form $h_b^\alpha(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ with $(i_j, S_{j,1}, S_{j,2}) \prec (i, S_1, S_2)$, it is easy

to see that $\mathcal{S} \vdash^* m'_j$ since $\mathcal{S} \vdash^* N_j, X_{k,j}, N'_j$. If m_j is not a handle, we also have $\mathcal{S} \vdash^* m'_j$. From all this, we deduce that $\mathcal{S} \vdash^* \{\!|m'_1, \ldots, m'_n|\!\}_{X_k}$ and the rule does not change the set of deducible terms. This proves the invariance of the induction hypothesis in this case.

We suppose in the following that $\mathcal{S} \nvdash^* X_k$ which allows us to apply Lemma 1 to $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{\{\!|m'_1, \ldots, m'_n|\!\}_{X_k}\}$.

- invariance of (**Sec***) : from Lemma 1 conclusion (1), if $h^\alpha_a(.,.,m,.,i,S'_1,S'_2) \in \mathsf{Handle}$ is such that $a \in H$, $S'_1, S'_2 \subseteq H$ and $\mathcal{S}' \vdash^* h^\alpha_a(.,.,m,.,i,S'_1,S'_2)$, then $\mathcal{S} \vdash^* h^\alpha_a(.,.,m,.,i,S'_1,S'_2)$ and the induction hypothesis allows us to conclude.

- invariance of (**Cert**) : ditto, from Lemma 1 conclusion (1), if

$$\mathcal{S}' \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$$

  with $a \in H$, $S'_1, S'_2 \subseteq H$ then $\mathcal{S} \vdash^* h_a(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i, S'_1, S'_2))$ and we conclude by using the induction hypothesis.

- invariance of (**SymEnc**) : From Lemma 1 conclusion (2) if $\mathcal{S}' \vdash^* \{\!|u|\!\}_k$ then either
  - $\mathcal{S} \vdash^* \{\!|u|\!\}_k$ and we conclude by using the induction hypothesis ;
  - or we have $\{\!|u|\!\}_k = \{\!|m'_1, \ldots, m'_n|\!\}_{X_k}$, which is built following the condition of the hypothesis.

- invariance of (**Sign**) : from Lemma 1 conclusion (3), $\mathcal{S}' \vdash^* \Sigma(u,k)$ and $\mathcal{S}' \nvdash^* k$ implies that $\mathcal{S} \vdash^* \Sigma(u,k)$ and the induction hypothesis allows us to conclude.

The rule (Sym Decrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$P_b(h^\alpha_b(N, \mathsf{Null}, X_k, \mathrm{symEncDec}, i, S_1, S_2)), P_b(\{\!|m_1, \ldots, m_n|\!\}_{X_k}) \Longrightarrow P_b(m'_1), \ldots, P_b(m'_n)$$

We remark that if $\mathcal{S} \vdash^* X_k$, then the rule creates handles and under which are stored terms which are already deducible from $\mathcal{S}$.

If $\mathcal{S} \nvdash^* X_k$, the by applying the hypothesis (**SymEnc**) to $\mathcal{S} \vdash^* \{\!|m_1, \ldots, m_n|\!\}_{X_k}$, we obtain that if m'_j is not a handle then $\mathcal{S} \vdash^* m'_j$. In all cases, the rule builds either terms which are deducible from $\mathcal{S}$, or handles. We can thus consider that the rule only creates handles. Let $h^\alpha_b(.,.,m,.,i_c,S'_1,S'_2) \in \mathsf{Handle}$ be such a handle. In order to be able to apply the Lemma 1, we want to show that if

$$\{b\} \cup S'_1 \cup S'_2 \not\subseteq H \tag{13}$$

then $\mathcal{S} \vdash^* m$. Indeed,

- if $\mathcal{S} \vdash^* X_k$ then $\mathcal{S} \vdash^* m$;

- if $\mathcal{S} \nvdash^* X_k$, then from property (**SymEnc**), there exists $c \in \mathsf{Agent}$ such that

$$\mathcal{S} \vdash^* h^\alpha_c(.,.,m,.,i_c,S'_1,S'_2).$$

  we know that $b \in H$ (the contrary would contradict $\mathcal{S} \nvdash^* X_k$). Furthermore, we have $S'_1 \cup S'_2 \not\subseteq H$ since we have $\{b\} \cup S'_1 \cup S'_2 \not\subseteq H$ and hence we deduce that $\mathcal{S} \vdash^* m$ which is what we want.

This allows us to use Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \mathsf{Hdls}$ where Hdls are the new handles created by the rule.

- invariance of (**Sec***) : let $h^\alpha_a(.,.,m,.,i,S'_1,S'_2) \in \mathsf{Handle}$ with $a \in H$, $S'_1, S'_2 \subseteq H$, from Lemma 1 conclusion (1), we have $\mathcal{S}' \vdash^* h^\alpha_a(.,.,m,.,i,S'_1,S'_2)$ with $a \in H$, $S'_1, S'_2 \subseteq H$, implies that either

- $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$ and we conclude by appealing to the induction hypothesis ;
- or $h_a^\alpha(.,.,m,.,i,S_1',S_2')$ is m_j' a handle created by applying the rule (**Sym Decrypt**) on $\{\!|m_1,\ldots,m_n|\!\}_{X_k}$.

In this last case, the handle m_j' is of the form $h_b^r(N_j,N_j',X_{k,j},T_j,i_j,S_{j,1},S_{j,2})$. We have to show that if $b \in H$ and $S_{j,1},S_{j,2} \subseteq H$ then $\mathcal{S}' \nvdash^* X_{k,j}$. We thus suppose that $b \in H$ and $S_{j,1},S_{j,2} \subseteq H$. As $(i_j,S_{j,1},S_{j,2}) \prec (i,S_1,S_2)$, by applying the hypothesis (**Sec***), we deduce that $\mathcal{S} \nvdash^* X_k$. Thus we can apply hypothesis (**SymEnc**) to $\mathcal{S} \vdash^* \{\!|m_1,\ldots,m_n|\!\}_{X_k}$. We obtain the existence of a $c \in H$ such that

$$\mathcal{S} \vdash^* h_c^\alpha(N,\mathsf{Null},X_k,\text{symEncDec},i_c,S_{c,1},S_{c,2})$$

and furthermore the existence of a handle $h_c(.,.,X_{k,j},.,i_j,S_{j,1},S_{j,2})$ with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$. We know that $\{c\} \cup S_{j,1} \cup S_{j,2} \subseteq H$ and by applying (**Sec***), we obtain that $\mathcal{S} \nvdash^* X_{k,j}$. Now, from conclusion (1) of Lemma 1, we deduce that $\mathcal{S}' \nvdash^* X_{k,j}$ and $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$ and that $b \in S_{j,2}$.

For the other hypothesis, properties (**Cert**), (**SymEnc**), (**Sign**) for $\mathcal{S}'$ reduces, by immediate application of respectively the conclusion (1), (2) and (3) of Lemma 1, to the same properties for $\mathcal{S}$.

The rule (Asym Gen) : Reminder of the rule (with the notations of Section 3.2) :

$$\begin{array}{l} P_e(h_e^\alpha(N_1,N_2,inv(Y_k),\text{privCertSign},i_1,S_{e,1},S_{e,2})),\\ \qquad P_e(h_e(N_2,\mathcal{C}(N_2,N_1,N_{cert},Y_k,\text{pubCertVerif},i_1,S_{e,1},S_{e,2}))),i_2,S_1,S_2,b \overset{N_3,N_4,X_k}{\Longrightarrow}\\ \qquad\quad P_e(h_e^g(N_3,N_4,inv(X_k),\text{privDecSign},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2)),\\ P_e(\Sigma(\mathcal{C}(N_4,N_3,N_2,X_k,\text{pubEncVerif},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2),inv(Y_k))) \end{array}$$

Let $S_{e,1}' = S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}$ and $S_{e,2}' = \{b,e\} \cup S_2$ be the control sets of the handle created by the rule. In the case that $\mathcal{S} \vdash^* inv(Y_k)$, then we have $\{e\} \cup S_{e,1} \cup S_{e,2} \nsubseteq H$ and thus $\{e\} \cup S_{e,1}' \cup S_{e,2}' \nsubseteq H$. Then, we apply the Lemma 1 to

$$\mathcal{S}_{\mathsf{int}}'' = \mathcal{S}_{\mathsf{int}} \cup \{h_e^g(N_3,N_4,inv(X_k),\text{privDecSign},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2)\} \cup \mathsf{NK},$$

where $\mathsf{NK} = \{N_3,N_4,X_k,inv(X_k)\}$. Indeed, it is clear that $N_3,N_4,X_k,inv(X_k)$ do not appear in $\mathcal{S}$ and furthermore $\{u|\mathcal{S}' \vdash^* u\} = \{u|\mathcal{S}'' \vdash^* u\}$.

In the case that $\mathcal{S} \nvdash^* inv(Y_k)$, we apply Lemma 1 to

$$\begin{array}{l} \mathcal{S}_{\mathsf{int}}' = \mathcal{S}_{\mathsf{int}} \cup \{h_e^g(N_3,N_4,inv(X_k),\text{privDecSign},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2)\} \cup\\ \qquad \{\Sigma(\mathcal{C}(N_4,N_3,N_2,X_k,\text{pubEncVerif},i_2,S_{e,1}\cup S_{e,2}\cup S_1\cup\{e\},\{b,e\}\cup S_2),inv(Y_k))\} \cup \mathsf{NK}, \end{array}$$

with $\mathsf{NK} = \{N_3,N_4,X_k\}$ if $\{e\} \cup S_{e,1}' \cup S_{e,2}' \subseteq H$ and else $\mathsf{NK} = \{N_3,N_4,X_k,inv(X_k)\}$.

- invariance of (**Sec***) : from Lemma 1 conclusion (1), we have that $\mathcal{S}' \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$ with $a \in H$, $S_1',S_2' \subseteq H$ implies either :
  - $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S_1',S_2')$;
  - or $h_a^\alpha(.,.,m,.,i,S_1',S_2')$ comes from the rule (**Asym Gen**).

  In the first case, we appeal to the induction hypothesis on $\mathcal{S}$. In the second case, we remark that $\{e\} \cup S_{e,1} \cup S_{e,2} \subseteq H$ and by applying (**Sec***), we obtain that $\mathcal{S} \nvdash^* inv(Y_k)$. As $inv(X_k) \notin \mathsf{Handle} \cup \{N_3,N_4,X_k\}$, Lemma 1 conclusion (1) gives $\mathcal{S}' \nvdash^* inv(X_k)$. Furthermore, it is clear that $inv(X_k) \in \mathsf{Keyv}$.

- invariance of (**Cert**) and of (**SymEnc**) : none of the terms concerned by these properties are created by this rule.
- invariance of (**Sign**) : if $\mathcal{S} \vdash^* inv(Y_k)$, thanks to the Lemma 1 conclusion (3) (with the same hypothesis as above), we easily obtain that a signature deducible from $\mathcal{S}'$ is already deducible from $\mathcal{S}$. If $\mathcal{S} \nvdash^* inv(Y_k)$, from Lemma 1 conclusion (3), we have $\mathcal{S}' \vdash^* \Sigma(t, inv(Y_k))$ implies either
  - $\mathcal{S} \vdash^* \Sigma(t, inv(Y_k))$;
  - or

    $$\Sigma(t, inv(Y_k)) = \Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubEncVerif}, i_2, S_{e,1} \cup S_{e,2} \cup S_1, \{b, e\} \cup S_2), inv(Y_k)).$$

  In the first case, the induction hypothesis allows us to conclude and in the second case, it is easily seen that the property (**Sign**) is verified.

The rule (Asym SignEncrypt) : Reminder of the rule (with the notations of Section 3.2) :

$$P_b(h_b^\alpha(N_1, N_2, inv(Y_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2}),$$
$$P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, X_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),$$
$$P_b(m_1), \ldots, P_b(m_n) \Longrightarrow P_b(\{m'_1, \ldots, m'_n\}_{X_k}), P_b(\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k)))$$

Suppose that $\mathcal{S} \vdash^* inv(Y_k)$, from the hypothesis (**Sec***) it means that $S_{b,1} \cup S_{b,2} \cup \{b\} \not\subseteq H$. Suppose now that for a j, we have $m_j = h_b^\alpha(N_j, N'_j, X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2})$ and $m'_j \neq \varnothing$. Then by construction, we have

$$(i_j, S_{j,1}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2}). \tag{14}$$

Thus, from the definition of $\vdash^*$ and because of (14), we have $\mathcal{S} \vdash^* N_j, X_{k,j}, N'_j$ and thus $\mathcal{S} \vdash^* m'_j$. Since furthermore, $\mathcal{S} \vdash^* m'_j$ if m_j is not a handle, we deduce that $\mathcal{S} \vdash^* \{m'_1, \ldots, m'_n\}_{X_k}$. We have moreover $\mathcal{S} \vdash^* \Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))$. In this case the execution of the rule does not change the set of deducible terms.

From now on, we suppose that $\mathcal{S} \nvdash^* inv(Y_k)$. Then, from the definition of $\vdash^*$, we have $\{b\} \cup S_{b,1} \cup S_{b,2} \subseteq H$. As by construction, we have $(i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2})$ (else the rule does nothing), by applying the hypothesis (**Cert**), there exists a $c \in S_{c,2}$ and a handle :

$$h_c^\alpha(N_4, N_3, inv(X_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$

By applying (**Sec***), we find that $\mathcal{S} \nvdash^* inv(X_k)$. As we have moreover, $\mathcal{S} \nvdash^* inv(Y_k)$, we can then use the Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{\{m'_1, \ldots, m'_n\}_{X_k}\} \cup \{\Sigma(\{m'_1, \ldots, m'_n\}_{X_k}, inv(Y_k))\}$.

- invariance of (**Sec***) : from the Lemma 1 conclusion (1), if there exists a handle $h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ such that

  $$\mathcal{S}' \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$$

  then $\mathcal{S} \vdash^* h_a^\alpha(., ., m, ., i, S'_1, S'_2)$ and the induction hypothesis allows us to conclude.
- invariance of (**Cert**) : ditto, the Lemma 1 conclusion (1) allows us to reduce directly to the induction hypothesis.
- invariance of (**SymEnc**) : the Lemma 1 conclusion (2) allows us to reduce to the induction hypothesis.
- invariance of (**Sign**) : by applying the Lemma 1 conclusion (3), we obtain that if $\Sigma(u, k)$ is a term such that $\mathcal{S}' \vdash^* \Sigma(u, k)$ then either $\mathcal{S} \vdash^* \Sigma(u, k)$ and the induction hypothesis allows us to conclude or $\Sigma(u, k)$ is generated by the rule and it is an immediate verification that the hypothesis (**Sign**) is fulfilled in this case.

The rule (Asym VerifDecrypt) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{aligned} &P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2}))),\\ &\qquad P_b(h_b^\alpha(N_4, N_5, inv(X_k), \text{privDecSign}, i_b, S_{b,1}, S_{b,2})),\\ &\qquad P_b(\{m_1, \ldots, m_n\}_{X_k}), P_b(\Sigma(\{m_1, \ldots, m_n\}_{X_k}, inv(Y_k)))\\ \Longrightarrow &P_b(m'_1), \ldots, P_b(m'_n) \end{aligned}$$

We first show that the terms created by the rule which are not handles are deducible from $\mathcal{S}$ so as to be able to use Lemma 1. We remark that if $\mathcal{S} \vdash^* inv(X_k)$ then all the m'_j which are not handles are deducible from $\mathcal{S}$ and we can consider that the execution of the rule only adds handles.

We suppose now that $b \in H$ (else $\mathcal{S} \vdash^* inv(X_k)$ which is the above case). If $\mathcal{S} \vdash^* inv(Y_k)$, we have $S_{c,1} \cup S_{c,2} \nsubseteq H$. Indeed, suppose on the contrary that $S_{c,1} \cup S_{c,2} \subseteq H$. By applying (**Cert**), we obtain the existence of a $c \in S_{c,2}$ and of a handle :

$$\mathcal{S} \vdash^* h_c^{\cdot}(N_2, N_1, inv(Y_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$

Now, from (**Sec***), we have $\mathcal{S} \nvdash^* inv(Y_k)$ which is a contradiction.

From the fact that $(i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2})$, we deduce $\mathcal{S} \vdash^* inv(X_k)$ which brings us back to the preceding case. If $\mathcal{S} \nvdash^* inv(Y_k)$, then the hypothesis (**Sign**) says that the m'_j which are not handles are deducible from $\mathcal{S}$.

From now on, we suppose that the execution of the rule only produces handles and terms which are deducible from $\mathcal{S}$. Let $h_b^\alpha(.,.,m,.,i,S'_1,S'_2)$ be such a handle. We are going to prove that if $\{b\} \cup S'_1 \cup S'_2 \nsubseteq H$ then $\mathcal{S} \vdash^* m$. Indeed,

- if $\mathcal{S} \vdash^* inv(X_k)$ this is clear since in this case we have $\mathcal{S} \vdash^* m_j$ for $j = 1 \ldots n$.
- if $\mathcal{S} \nvdash^* inv(X_k)$, we deduce from the definition of $\nvdash^*$ that $\{b\} \cup S_{b,1} \cup S_{b,2} \subseteq H$. As $(i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2})$, we can apply (**Cert**) which gives us the existence of a $c \in S_{c,2}$ and a handle such that :

  $$\mathcal{S} \vdash^* h_c^{\cdot}(N_2, N_1, inv(Y_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$

  As, we have $\{c\} \cup S_{c,1} \cup S_{c,2} \subseteq H$, from (**Sec***), we have that $\mathcal{S} \nvdash^* inv(Y_k)$. We can then apply (**Sign**) which gives us the existence of a handle

  $$h_d^\alpha(.,.,m,.,i,S'_1,S'_2),$$

  for $d \in \mathsf{Agent}$. But as $S'_1 \cup S'_2 \nsubseteq H$ (else $b \notin H$ and this is a contradiction with the fact that $\mathcal{S} \nvdash^* inv(X_k)$), we deduce that $\mathcal{S} \vdash^* m$.

We can summarise all the above by saying that we can suppose that the execution of the rule only produce handles or terms which are deducible from $\mathcal{S}$ and that if the value of these handles is deducible from $\mathcal{S}'$ then it was already deducible from $\mathcal{S}$. Let Hdls be the set of handles created by the rule, we can thus apply the Lemma 1 with $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \mathsf{Hdls}$.

- invariance of (**Sec***) : let $h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$ be a handle with $a \in H$, $S'_1, S'_2 \subseteq H$, from the Lemma 1 conclusion (1), we have that $\mathcal{S}' \vdash^* h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$, implies either :
  - $\mathcal{S} \vdash^* h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$ and the induction hypothesis allows us to conclude;
  - or $h_a^\alpha(.,.,m,.,i,S'_1,S'_2)$ is a handle created by application of the rule (**Asym VerifDecrypt**) on $\{m_1, \ldots, m_n\}_{X_k}$.

In this case, we have

$$h_a^\alpha(.,.,m,.,i,S_1',S_2') = h_b^r(N_j, N_j', X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}),$$

with

$$(i_j, S_{j,2}, S_{j,2}) \prec (i_b, S_{b,1}, S_{b,2}) \preceq (i_c, S_{c,1}, S_{c,2}). \tag{15}$$

We have to prove that if $b \in H$ and $S_{j,1}, S_{j,2} \subseteq H$ then $\mathcal{S}' \nvdash X_{k,j}$. Since

$$\mathcal{S} \vdash^* h_b(N_1, \mathcal{C}(N_1, N_2, N_3, Y_k, \text{pubEncVerif}, i_c, S_{c,1}, S_{c,2})),$$

from the hypothesis (**Cert**) and (15), we know that $\exists c \in S_{c,2}$ such as

$$\mathcal{S} \vdash^* h_c^{\cdot}(N_2, N_1, inv(Y_k), \text{privDecSign}, i_c, S_{c,1}, S_{c,2}).$$

From (15), we have $c \in H$ and $S_{c,1}, S_{c,2} \subseteq H$ so that by application of (**Sec***), we obtain that

$$\mathcal{S} \nvdash^* inv(Y_k). \tag{16}$$

Now, thanks to (**Sign**) and (16), we obtain the existence of a handle

$$\mathcal{S} \vdash^* h_d^{\cdot}(N_j, N_j', X_{k,j}, T_j, i_j, S_{j,1}, S_{j,2}),$$

with $X_{k,j} \in \mathsf{Keyv} \cup \mathsf{Noncev}$. As $d \in H$, $S_{j,1}, S_{j,2} \subseteq H$, by applying (**Sec***), we obtain that $\mathcal{S} \nvdash^* X_{k,j}$, $X_{k,j} \in \mathsf{Key} \cup \mathsf{Nonce}$ and $b \in S_{j,2}$.

The other hypothesis can be proved immediately by reduction over the induction hypothesis via the Lemma 1 with $\mathcal{S}' = \mathcal{S} \cup \mathsf{Hdls}$.

The rule (Cert Gen) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{array}{l} P_e(h_e^\alpha(N_1, N_2, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2})), \\ \qquad P_e(h_e(N_2, \mathcal{C}(N_2, N_1, N_{cert}, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))), i_b, S_1, S_2 \overset{N_3, N_4, X_k}{\Longrightarrow} \\ \qquad P_e(h_e^g(N_3, N_4, inv(X_k), \text{privCertSign}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2)), \\ P_e(\Sigma(\mathcal{C}(N_4, N_3, N_2, X_k, \text{pubCertVerif}, i_b, S_{e,1} \cup S_{e,2} \cup S_1 \cup \{e\}, \{e, b\} \cup S_2), inv(Y_k))) \end{array}$$

The proof for this rule is exactly the same as that of (**Asym Gen**).

The rule (Cert Verif) : Reminder of the rule (with the notations of the Section 3.2) :

$$\begin{array}{l} P_b(\Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k))), \\ \qquad P_b(h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, \text{pubCertVerif}, i_e, S_{e,1}, S_{e,2}))) \Longrightarrow \\ P_b(h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))) \end{array}$$

The rule creates a handle the value of which is clearly deducible from $\mathcal{S}$ (because it is inside the signature). We can thus apply the 1, by setting $\mathcal{S}'_{\mathsf{int}} = \mathcal{S}_{\mathsf{int}} \cup \{h_b(N_1, \mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}))\}$.

- invariance of (**Sec***) : it reduces immediately to the induction hypothesis thanks to the Lemma 1 conclusion (1).

- invariance of (**Cert**) : because of Lemma 1 conclusion (1) and the induction hypothesis we are reduced to proving (**Cert**) for a integrity handle produced by the execution of the rule thus that if $b \in H$, $S_{c,1} \subseteq H$ are such that

$$(i_c, S_{c,1}, \varnothing) \prec (i_e, S_{e,1} \cup S_{e,2}, \varnothing), \tag{17}$$

then there exists $S_{c,2} \subseteq H$ and $c \in S_{c,2}$ such that :

$$\mathcal{S}' \vdash^* h_c^{\cdot}(N_2, N_1, inv(X_k), priv\Theta, i_c, S_{c,1}, S_{c,2}). \tag{18}$$

By applying the hypothesis (**Cert**) to

$$\mathcal{S} \vdash^* h_b(N_3, \mathcal{C}(N_3, N_4, N_5, Y_k, S_{e,1}, \text{pubCertVerif}, i_e, S_{e,2})),$$

we deduce the existence of $e \in S_{e,2}$ such that

$$\mathcal{S} \vdash^* h_e^{\cdot}(N_4, N_3, inv(Y_k), \text{privCertSign}, i_e, S_{e,1}, S_{e,2}).$$

Because of (17), we have $e \in H$ and $S_{e,1} \cup S_{e,2} \subseteq H$, so that from (**Sec***), $\mathcal{S} \nvdash^* inv(Y_k)$. By applying (**Sign**) to

$$\mathcal{S} \vdash^* \Sigma(\mathcal{C}(N_1, N_2, N_3, X_k, pub\Theta, i_c, S_{c,1}, S_{c,2}), inv(Y_k)),$$

we deduce (18) with $c \in S_{c,2}$.

- invariance of (**SymEnc**) : ditto the rule (**Sym Gen**).
- invariance of (**Sign**) : ditto the rule (**Sym Encrypt**).

□

References

[1] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter Mosses, and Takayasu Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer Berlin / Heidelberg, 2000.

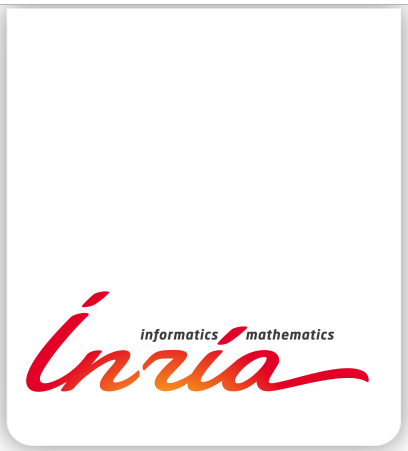
[2] C. Cachin and N. Chandran. A secure cryptographic token interface. In *Computer Security Foundations (CSF-22)*, pages 141–153, Long Island, New York, 2009. IEEE Computer Society Press.

[3] J. Clulow. On the security of PKCS#11. In *Proceedings of CHES 2003*, pages 411–425, 2003.

[4] Véronique Cortier and Graham Steel. A generic security api for symmetric key management on cryptographic devices. In Michael Backes and Peng Ning, editors, *Computer Security - ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 605–620. Springer Berlin / Heidelberg, 2009.

[5] Steve Kremer, Robert Knnemann, and Graham Steel. Universally composable key-management. *IACR Cryptology ePrint Archive*, 2012:189, 2012.

[6] Steve Kremer, Graham Steel, and Bogdan Warinschi. Security for key management interfaces. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11)*, pages 266–280, Cernay-la-Ville, France, June 2011. IEEE Computer Society Press.



RESEARCH CENTRE
PARIS – ROCQUENCOURT

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399