Unité Mixte de Recherche 5104 CNRS - INPG - UJF
Centre Equation
2, avenue de VIGNATE
F-38610 GIERES
tel : +33 456 52 03 40
fax : +33 456 52 03 50
http://www-verimag.imag.fr

Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries

R. Janvier, Y. Lakhnech and L. Mazaré

Report n^o 19

January 28, 2005

Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries

R. Janvier, Y. Lakhnech and L. Mazaré

VERIMAG
2, av. de Vignates, 38610 Gières - FRANCE
{romain.janvier,yassine.lakhnech,laurent.mazare}@imag.fr

January 28, 2005

Abstract

In this paper, we extend previous results relating the Dolev-Yao model and the computational model. We add the possibility to exchange keys and consider cryptographic primitives such as signature. This work can be applied to check protocols in the computational model by using automatic verification tools in the formal model. To obtain this result, we introduce a precise definition for security criteria which leads to a nice reduction theorem. The reduction theorem is of interest on its own as it seems to be a powerful tool for proving equivalences between security criteria. Also, the proof of this theorem uses original ideas that seem to be applicable in other situations.

Keywords: Security, Cryptographic Protocols, Formal Encryption, Probabilistic Encryption, Dolev-Yao Model, Computational Model.

Reviewers: Radu Iosif

Notes: *A totally revamped version of this document appears as technical report [21]. Results given in the latter are more general and the formalization of "a criterion" is probably easier to understand. Therefore, we encourage readers to consult [21] instead of this document.*

How to cite this report:

```
@techreport { verimag-TR-2004-19,
title = { Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries},
authors = { R. Janvier, Y. Lakhnech and L. Mazaré },
institution = { Verimag Technical Report },
number = {19},
year = { 2004},
note = { }
}
```

1 Introduction

Historically, verification of cryptographic protocols has been separated in two distinct branches. The first one, formal verification of security protocols, originates from the work of Dolev and Yao and was first described in [11]. The main hypothesis, called perfect cryptography assumption, is an abstraction of reality: the intruder can gain information on an encoded message only if he knows the inverse of the key used to create the message. Even with this strong assumption, flaws have been found in protocols that were believed to be secure (the most famous one has been exposed by G. Lowe in [17], some of them are listed in [8]). Under this assumption, automatic verification of security protocols is possible [22, 7]or [23, 14, 4](even if it needs other abstractions when considering an unbounded number of sessions) and has been successfully implemented in tools such as those proposed by project EVA [24]. However, the perfect cryptography assumption needs valid foundations and this is why recent works tried to weaken this hypothesis, either by adding equational theories [9], [1]or by other modifications of the Dolev-Yao model, adding guessing attacks for example [18]. In the second approach, encryption schemes are studied using a computational model based on Turing machines. In this context, there is no idealization made concerning the cryptographic schemes: cryptographic functions operate on strings, attackers are Turing machines and correctness is defined in terms of high complexity and weak probability of success [12, 3]. This computational approach is recognized as more realistic than the formal approach. However, its complexity makes it very difficult to develop (semi-)automatic verification methods.

In the last years, attempts have been made to bridge the gap between these two approaches. The ultimate objective is to be able to prove security in the formal model, then to prove properties on the encryption scheme and with that to deduce security of the protocol in the computational model. These first works successfully proved this kind of composition properties. Very restrictive hypothesis have been made to deal with the complexity of the computational model (as for example in the case of [20]). The first paper in this recent trend [2] proved that a notion of indistinguishability in the formal model is valid in the computational model when making some assumptions on the encryption scheme. This means that if two messages are not distinguishable in the formal model, then their computational equivalent cannot be separated by a Turing machine in a reasonable (polynomial) time. This work has been pushed further in [25] and then in [20]. This last paper proves that if the encryption scheme verifies a certain property (called IND-CCA), then security in the formal model implies security in the computational model. The important part of this work is that it works in the case of active adversaries and so can eventually be applied to protocols working on an insecure network like Internet. Our objective in this paper is to continue this work and release some of the strong hypothesis made over protocols. Other related works include Backes, Pflizmann and Waidner [19]. In this paper, the formal model is not exactly the Dolev-Yao model, although very close. It is not clear whether protocols can be checked automatically in this formalism. Also, the cryptographic primitives are modeled at a rather detailed level in the computational model. P. Laud [16] proves safety of the formal model for symmetric encryption. In particular, he deals with encryption cycles.

Our objective in this paper is to continue this work and weaken some of the restrictions imposed on protocols in previous works. The main restriction in [20] is that secret keys cannot be part of sent messages and that message forwarding is not allowed. To weaken these restrictions, we first give a general definition of a security criterion (like IND-CCA). These criteria can be seen as a game that an intruder should not be able to win. Our first result is a reduction theorem that proves the equivalence between a criterion and simpler criteria. This allows us to prove that the IND-CCA criterion is equivalent to quite richer and useful criteria. Definition of criteria is an important part as they make it possible to release some of the restrictions over protocols made by previous works. Finally, we use these criteria in order to prove that Dolev-Yao constitutes a safe abstraction of the computational model even for protocols involving both asymmetric encoding and digital signature. The next section gives the necessary definitions for using both the computational model and the formal model. The main point in this section is the definition of polynomial random Turing machine that can access oracles. A general definition for security criteria is given in the third section. We also formulate the reduction theorem and give its proof in this section. The following section applies this theorem to prove classical and new results. In section 5, the former reductions are used to prove that when the IND-CCA assumption holds, properties proved in [20] can be extended to a wider class of protocols. Thus, this section prove that verification in the Dolev-Yao model is sound in the computational model. Finally, section 6 conclude this paper.

2 Preliminaries

2.1 Definitions for the Computational Model

An *asymmetric encryption scheme* $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined by three algorithms. The key generation algorithm $\mathcal{KG}$ is a randomized function which given a security parameter η outputs a pair of keys (pk, sk), where pk is a public key and sk the associated secret key. The encryption algorithm $\mathcal{E}$ is also a randomized function which given a message and a public key outputs the encryption of the message by the public key. Finally the decryption algorithm $\mathcal{D}$ takes as input a cyphertext and a secret key and outputs the corresponding plain-text, i.e., $\mathcal{D}(\mathcal{E}(m, pk), sk) = m$. The execution time of the three algorithms is assumed polynomially bounded by η.

A *signature scheme* $\mathcal{SS} = (\mathcal{KG}, \mathcal{S}, \mathcal{V})$ is also defined by three algorithms. The key generation algorithm randomly generates pairs of keys (sik, vk), where sik is the signature key and vk is the verification key. The signature algorithm $\mathcal{S}$ randomly produces a signature of a given message by a given signature key. The verification algorithm $\mathcal{V}$ is given a message m, a signature σ and a verification key vk and tests if σ is a signature of m with the signature key corresponding to vk. Hence, $\mathcal{V}(m, \mathcal{S}(m, sik), vk)$ returns true for any message m and any pair of keys (sik, vk) generated by $\mathcal{KG}$. In this case, we still assume that the algorithms have a polynomial complexity.

Randomized Turing Machines with Oracles

An adversary for a given scheme is a Polynomial Random Turing Machine (PRTM) which has access to a set of oracles. These oracles depend on the scheme and are given in the different cases thereafter. We consider Turing machines which execution is polynomially bounded in the security parameter η, i.e. for any input corresponding to security parameter η, the machine stops within $P(\eta)$ steps for some polynomial P.

To model access to oracles, we slightly modify the definition of Turing machines. Our Turing machines have two additional tapes, one for arguments (of function/oracle calls) and one for results. Then, let F be a countable set of function names. We define our PRTM as a pair of a Turing machine $\mathcal{A}$, where transitions can be function calls, and a substitution σ linking function names $\underline{f} \in F$ to functions from string of bits (arguments) to string of bits (results). These functions are also described by polynomial Turing machines (which can also access oracles). To distinguish oracles from real functions (which can be their implementations), function names are always underlined when considering access to an oracle. The semantics of $\mathcal{A}/\sigma$ are the standard semantics of $\mathcal{A}$ except that whenever $\mathcal{A}$ fires a transition labeled by a function call $\underline{f}$, the content of the results tape becomes $f\sigma(args)$, where $args$ is the value of the arguments tape.

To simplify notations, we write directly $\mathcal{A}/f_1, ..., f_n$ where f_i are functions. Thus, we omit the name of the function as soon as this name is not relevant for comprehension. Functions are directly called using the $\underline{f_i}$ notation when defining $\mathcal{A}$.

A function $h : \mathbb{R} \to \mathbb{R}$ is *negligible*, if it is ultimately bounded by x^{-c}, for each positive $c \in \mathbb{N}$, i.e., for all $c > 0$ there exists N_c such that $|h(x)| < x^{-c}$, for all $x > N_c$.

The definition of messages and of the intruder in the formal model is by now standard, e.g. [11, 22].

2.2 Definitions for the Formal Model

In this section, we give the basic definitions that will be used to introduce the formal aspects of protocol checking. Formal studies rely on the concept of messages which are first order terms. To define messages, we first introduce three infinite disjoint sets : $Nonces$, $Identity$ and $Keys$. Elements of $Nonces$ will usually be denoted by N and can be thought as random numbers. Thus, it is impossible for an intruder to guess the value of a nonce without indications. Elements of $Identity$ are the possible names of agents involved in the protocol. Finally, elements of $Keys$ represent asymmetric encryption keys. There is a unary function over $Keys$ associating each key k to its inverse k^{-1} such that $k = (k^{-1})^{-1}$. Two binary operators are defined over messages: concatenation and encryption. Concatenation of messages m and n will be written $\langle m, n \rangle$. Encryption of message m with key k will be denoted by $\{m\}_k$.

The *entailment* relation $E \vdash m$ (m is deducible from E) is defined as in [11] when E is a finite set of messages and m a message. This relation is defined as the least binary relation verifying:

- If $m \in E$, then $E \vdash m$.
- If $E \vdash m$ and $E \vdash n$, then $E \vdash \langle m, n \rangle$.
- If $E \vdash \langle m, n \rangle$, then $E \vdash m$.
- If $E \vdash \langle m, n \rangle$, then $E \vdash n$.
- If $E \vdash m$ and $E \vdash k$, then $E \vdash \{m\}_k$.
- If $E \vdash \{m\}_k$ and $E \vdash k^{-1}$, then $E \vdash m$.

Note that symmetric encryption can be represented using keys k such that $k^{-1} = k$ and signature can be represented by encoding with a private key to sign and decoding with the related public key to verify the signature.

3 A Generic Reduction Theorem

In [20], protocols allowing sending of secret keys are not considered because it is not possible in IND-CCA to encode secret keys. To solve that, we introduce a new criterion N-PAT-IND-CCA and prove it equivalent to IND-CCA. A similar result is needed to introduce signature.

A security criterion γ is defined by an experiment that involves an adversary and two ways W_0 and W_1 of implementing a set of oracles. The adversary is aware of both implementations and is allowed to call the oracles but does not know which implementation is really used. The challenge consists in guessing which implementation is used. More precisely, an adversary is a probabilistic polynomial time Turing machine (PRTM) that has access to a set of oracles (either W_0 or W_1). The adversary's *advantage* is the probability that the adversary outputs 1 when the set of oracles is W_1 minus the probability that the adversary outputs 1 when the set of oracles is W_0. An encryption scheme is said γ-secure, if the advantage of any adversary is negligible.

In this section, we present a generic result allowing us to prove that a security criterion γ_1 can be reduced to a criterion γ_2. This means that if there exists an adversary that breaks γ_2 then there exists an adversary that breaks γ_1. The proof is constructive in the sense that such an adversary for γ_1 can be effectively computed.

Given a finite set x_i where i ranges from 1 to n, $\overline{x}$ denotes the whole set of x_i. When more precision is required, this set can also be denoted by $x_{1..n}$. In this section, we give a formal definition of a criterion and show how a criterion can be partitioned in a safe way. The theorem presented here allows us to verify that a criterion is equivalent to another one by using such partitions. This result is applied in the following sections to show an equivalence between a few security criteria.

3.1 Security Criterion

A criterion γ is a collection formed by:

- A bit b, this bit is the challenge that has to be guessed by the adversary.
- A finite number of parameters c_1 to c_{na}. These parameters are shared by the oracles and most of the time, they are chosen randomly at the beginning of the experiment. Θ is the PRTM producing these parameters (usually a key generation algorithm).
- A finite number of oracles f_1 to f_{nb} that depend on their argument, $\overline{c}$ and b. For each f_i, there exist f_i^α and f_i^β such that the corresponding oracles when given argument (l, r) produce $f_i^\beta(f_i^\alpha(l, \overline{c}), \overline{c})$ when $b = 0$ and $f_i^\beta(f_i^\alpha(r, \overline{c}), \overline{c})$ when $b = 1$.
- A finite number of oracles g_1 to g_{nc} that depend on their argument and $\overline{c}$. The corresponding oracles when given argument x produce $g_i(x, \overline{c})$.

Oracles in $\overline{g}$ do not depend on b, they cannot be used directly by the adversary to gain information on b but they can be useful by giving information on the shared parameters $\overline{c}$ that can finally allow the adversary to deduce the value of b. Oracles in $\overline{f}$ have two layers α and β, these layers are used to decompose a criterion into a partition of criteria as the α layer allows the β layers to depend on less parameters.

Example 3.1 *Let γ be the criterion $(b, \{pk_1, sk_1, pk_2, sk_2\}, \{f_1, f_2\}, \emptyset)$. Functions f_1 and f_2 have no α layer, i.e. $f_i^\alpha(x, ...) = x$ and $f_i(m_0, m_1)$ corresponds to the encryption of message m_b using key pk_i. Thus γ corresponds to 2-IND-CPA as introduced in [5]: the adversary has to guess the value of bit b by using only two oracles that encrypt the left or the right message according to b. The 2-IND-CCA criterion can be obtained by adding two oracles g_1 and g_2. These oracles decrypt messages encoded respectively with key pk_1 and pk_2 assuming that these messages have not been produced by oracle f_1 or f_2.*

The advantage of a PRTM $\mathcal{A}$ against γ is

$$\mathbf{Adv}^\gamma_{\mathcal{A}}(\eta) = Pr[\mathbf{Exp}^\gamma_{\mathcal{A}}(\eta, 1) = 1] - Pr[\mathbf{Exp}^\gamma_{\mathcal{A}}(\eta, 0) = 1]$$

Where Exp is the Turing machine defined by:

Experiment $\mathbf{Exp}^\gamma_{\mathcal{A}}(\eta, b)$:
$\overline{c} \xleftarrow{R} \Theta(\eta)$
if $b = 0$ **then**
  $f_i \leftarrow \lambda(l, r).f_i^\beta(f_i^\alpha(l, \overline{c}), \overline{c})$ for i in $1...nb$
else
  $f_i \leftarrow \lambda(l, r).f_i^\beta(f_i^\alpha(r, \overline{c}), \overline{c})$ for i in $1...nb$
$d \xleftarrow{R} \mathcal{A}/\eta, \overline{f}, \overline{g}$
return d

$\mathcal{A}$ has access to an oracle giving η and to the oracles $\overline{f}$ and $\overline{g}$ as defined above. Oracles in $\overline{f}$ depend on b, this dependence is explicited by "creating" oracles f according to the value of b.

The advantage of $\mathcal{A}$ is the probability to answer 1 when the value of b is 1 minus the probability to answer 1 when the value of b is 0. Thus, if we consider a machine $\mathcal{A}$ that always outputs the same result, its advantage is 0, this also holds when considering a machine that gives a random output. Advantages are between -1 and 1, however, if $\mathcal{A}$ has a negative advantage, it is easy to build a PRTM $\mathcal{B}$ that has the opposite of $\mathcal{A}$'s advantage (we simply need to run $\mathcal{A}$ and to return the inverse of its output).

3.2 Criterion Partition and the Reduction Theorem

Example 3.2 *Let us consider the 2-IND-CPA criterion γ defined before. Then, we say that $\gamma' = (b, \{pk_1, sk_1\}, \{f_1\}, \emptyset)$ and $\gamma'' = (b, \{pk_2, sk_2\}, \{f_2\}, \emptyset)$ constitutes a valid partition of γ when both criteria are valid (i.e. f_1 and f_2 are in the same criterion as their respective parameters pk_1 and pk_2). γ' and γ'' correspond to the IND-CPA criterion (only one oracle is available). By the reduction theorem, if an encryption scheme is IND-CPA secure (advantage of any PRTM against γ' and γ'' is negligible), then it is 2-IND-CPA secure.*

A pair of criteria γ', γ'' defines a *valid partition* of γ if there exist na', nb' and nc' such that

- $\gamma' = \left(b, c_{1..na'}, f^\beta_{1..nb'}, g_{1..nc'}\right)$
- $\gamma'' = \left(b, c_{(na'+1)..na}, f_{(nb'+1)..nb}, g_{(nc'+1)..nc}\right)$
- For $i \leq nb'$, f_i^α only depends on $c_{(na'+1)..na}$.
- For $i \leq nb'$, f_i^β only depends on $c_{1..na'}$.
- For $i \leq nc'$, g_i only depends on $c_{1..na'}$.

- For $i > nb'$, f_i only depends on $c_{(na'+1)..na}$.
- For $i > nc'$, g_i only depends on $c_{(na'+1)..na}$.

The four last conditions are necessary for γ' and γ'' to remain valid: oracles from a criterion only have access to parameters generated by the same criterion. The reduction theorem states that an advantage against a criterion γ can be used to produce an advantage over criterion γ' or criterion γ''.

Theorem 3.1 (Reduction Theorem) *If γ', γ'' is a valid partition of γ and $\mathcal{A}$ is a PRTM, then there exist two PRTM $\mathcal{A}^o$ and $\mathcal{B}$ such that*

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma'}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma''}_{\mathcal{A}^o}(\eta)|$$

Proof Idea for the Reduction Theorem

The purpose of this section is to explain the main ideas underlying the proof of the reduction theorem, the detailed proof appears in appendix B . An application of this proof to a simple example is given below.

The adversary $\mathcal{A}^o$ against the criterion γ'' simulates $\mathcal{A}$. To do so, he has to answer the queries made to oracles from γ'. Since $\mathcal{A}^o$ cannot construct faithfully these oracles (as it does not have access to parameters from γ''), it returns incorrect answers to $\mathcal{A}$. Finally $\mathcal{A}^o$ uses the output of $\mathcal{A}$ to answer its own challenge. If the advantage of $\mathcal{A}^o$ is comparable to the advantage of $\mathcal{A}$, then γ can be reduced to criterion γ''.

Else the advantage of $\mathcal{A}^o$ is negligible compared to the advantage of $\mathcal{A}$, then another adversary, $\mathcal{B}$, has the same advantage as $\mathcal{A}$. The adversary $\mathcal{B}$ is playing against the criterion γ'. It generates a challenge for $\mathcal{A}$. Moreover, if $b = 1$ the answers to the queries made by $\mathcal{A}$ are correct and if $b = 0$ the answers are forged in the same way as in $\mathcal{A}^o$. When $\mathcal{A}$ answers its challenge, $\mathcal{B}$ verifies it. If it is correct, $\mathcal{B}$ supposes that $b = 1$, else it supposes that $b = 0$. Indeed, $\mathcal{A}^o$ probably has a lower advantage than $\mathcal{A}$.

Example 3.3 *Consider our previous (IND-CPA) example. Machine $\mathcal{A}^o$ is opposed to γ'', it creates the missing key pk_1 and uses it to simulate the missing oracle: the simulation is achieved by always encoding the left argument, ${}_{fake}f_1(l, r) = \mathcal{E}(l, pk_1)$. Construction of $\mathcal{B}$ is detailed in figure 1. Machine $\mathcal{B}$ is opposed to γ' with the*

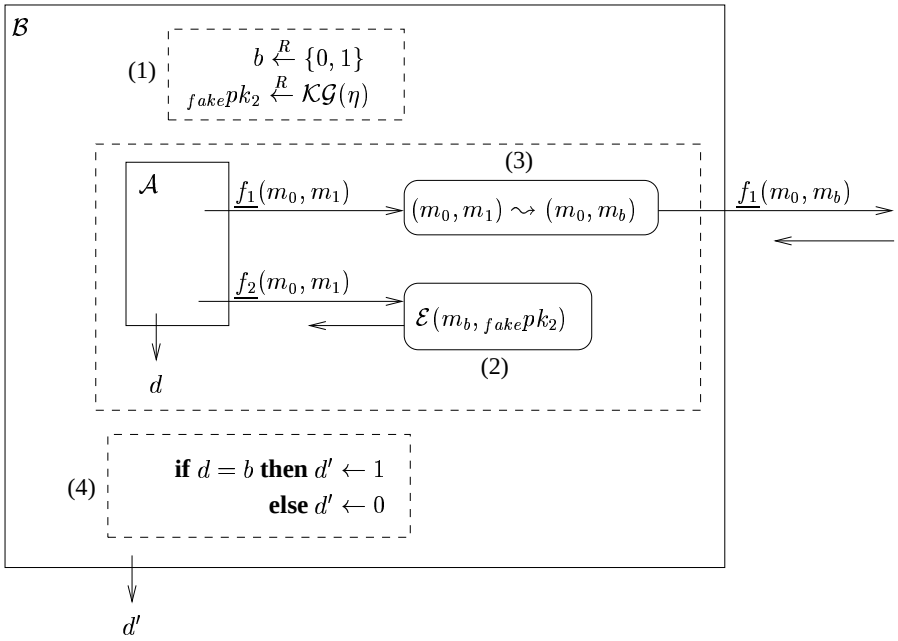

Figure 1: Behavior of $\mathcal{B}$

challenge bit b'. It creates its missing key pk_2 and a random bit b(1). The fake oracle ${}_{fake}f_2$ uses this bit b(2). Oracle f_1 is also faked using b': ${}_{fake}f_1(m_0, m_1) = f_1(m_0, m_b)$. The faked oracles behave like the original oracles when $b' = 1$. They behave like the oracles faked in $\mathcal{A}^o$ when $b' = 0$. This behavior is summed up in the following array:

$oracles$	$b' = 0$	$b' = 1$
$\underline{\mathcal{E}_{pk_1}}(m_0, m_1)$	$\mathcal{E}(m_0, pk_1)$	$\mathcal{E}(m_b, pk_1)$
$\underline{\mathcal{E}_{pk_2}}(m_0, m_1)$	$\mathcal{E}(m_b, pk_2)$	$\mathcal{E}(m_b, pk_2)$

Then, if the underlying $\mathcal{A}$ machine answers b correctly, we assume that it was confronted to the right oracles and thus machine $\mathcal{B}$ answers 1, else it answers 0(4). The intuition behind machine $\mathcal{B}$ is that its advantage tells whether oracles from γ' are useful or can be faked without losing the advantage.

4 Applications of the reduction theorem

4.1 Reducing N-IND-CCA to IND-CCA

The first application of our theorem will concern an already proved result: reducing N-IND-CCA to IND-CCA. A proof of this result can be found in [5].

Let $\mathcal{AE}$ be an asymmetric encryption scheme. We define the security criterion N-IND-CCA, for $N \geq 1$, where N is the number of pairs of public/private keys that form the challenge, i.e. an attacker will try to break one of these private keys. Thus, an attacker is a PRTM $\mathcal{A}$ that has access to oracles corresponding to the N encryption and N decryption functions associated to the N pairs of keys. When $N = 1$, we simply write IND-CCA.

An N-IND-CCA adversary has access to the N *left-right encryption oracles* $\mathcal{E}_{pk_i}(LR(.,.,b))$, which given two messages m_0 and m_1 output the encryption of m_b by pk_i (LR is defined by $LR(m_0, m_1, b) = m_b$). The adversary must guess the value of bit b. That is, the first implementation corresponds to $b = 0$ and the second to $b = 1$. The adversary is also given N decryption oracles $\mathcal{D}_{sk_i}(.)$ which cannot be used on cyphertexts produced by the left-right encryption oracles.

Thus, the adversary plays against a criterion γ_N including a bit b , parameters $\overline{c}$ are the pair of keys (pk_i, sk_i), oracles in $\overline{f}$ are $\mathcal{E}_{pk_i}(LR(.,.,b))$ and oracles in $\overline{g}$ are $\mathcal{D}_{sk_i}(.)$ where i ranges from 1 to N. Oracles in $\overline{f}$ have no α layer (i.e. f_i^α is the identity).

An asymmetric encryption scheme $\mathcal{AE}$ is said to be N-IND-CCA iff for all adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}_{\mathcal{A}}^{\gamma_N}(\eta)$ is negligible.

Lemma 4.1 *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is N-IND-CCA, then $\mathcal{AE}$ is also IND-CCA.*

Proof: Let $\mathcal{A}$ be an IND-CCA adversary. We can easily construct a N-IND-CCA adversary $\mathcal{B}$ such that : $|\mathbf{Adv}_{\mathcal{A}}^{\gamma_1}(\eta)| = |\mathbf{Adv}_{\mathcal{B}}^{\gamma_N}(\eta)|$. Hence if $\mathcal{AE}$ is N-IND-CCA, the N-IND-CCA advantage (related to γ_N) of $\mathcal{B}$ is negligible and so is the IND-CCA advantage of $\mathcal{A}$. ■

Proposition 4.1 *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is N-IND-CCA, then $\mathcal{AE}$ is also (N+1)-IND-CCA.*

Proof: Criterion γ_{N+1} has a valid partition constituted by γ_1 and γ_N.

The reduction theorem applies and gives:

$$|\mathbf{Adv}_{\mathcal{A}}^{\gamma_{N+1}}(\eta)| \leq 2.|\mathbf{Adv}_{\mathcal{B}}^{\gamma_1}(\eta)| + |\mathbf{Adv}_{\mathcal{A}^o}^{\gamma_N}(\eta)|$$

By hypothesis, $\mathcal{AE}$ is N-IND-CCA (hence IND-CCA). Then the advantages of $\mathcal{B}$ and $\mathcal{A}^o$ are negligible and we can conclude that the advantage of $\mathcal{A}$ is negligible too. ■

Corollary 4.1 *For any N, $\mathcal{AE}$ is IND-CCA if and only if $\mathcal{AE}$ is also N-IND-CCA.*

Above, for simplicity's sake, the adversary does not have access to the public keys whereas he should in the classical IND-CCA definition. However, it is easy to modify the previous definitions by adding oracles in $\overline{g}$ that will just output the public keys.

This result can be extended to an unbounded number of keys that is still polynomial in η [21].

4.2 Adding Patterns: N-PAT-IND-CCA

We now introduce a new security criterion and prove it equivalent to IND-CCA by using our reduction theorem. N-PAT-IND-CCA allows the adversary to obtain the encryption of messages containing challenge secret keys, even if it does not know the value of these secret keys. For that purpose, the adversary is allowed to give pattern terms to the left-right oracles.

The pattern terms are terms where new atomic constants have been added: pattern variables. These variables denote the different challenge secret keys ($[i]$ asks the oracle to replace it with the value of sk_i). Variables can be used as atomic messages (data pattern). When a left-right oracle is given a pattern term, it replaces patterns by values of corresponding keys and encodes the message. Formally, patterns are given by the following grammar where bs is a bit-string and i is an integer.

$$pat \quad ::= \quad \langle pat, pat\rangle \,|\, \{pat\}_{bs} \,|\, bs \,|\, [i]$$

The computation (valuation) made by the oracle is easily defined recursively in a context giving the bit-string values for the different keys. Its result is a bit-string and it uses the encryption algorithm $\mathcal{E}$ and the concatenation denoted by operator $\cdot$.

$$v(bs, \overline{pk, sk}) = bs \qquad v(\{p\}_{bs}, \overline{pk, sk}) = \mathcal{E}(v(p, \overline{pk, sk}), bs))$$
$$v([i], \overline{pk, sk}) = sk_i \qquad v(\langle p_1, p_2\rangle, \overline{pk, sk}) = v(p_1, \overline{pk, sk}).v(p_2, \overline{pk, sk})$$

There is yet a restriction: we exclude encryption cycles. Hence keys are ordered and a pattern $[i]$ can only be encrypted under pk_j if $i > j$. Without this restriction N-PAT-IND-CCA is strictly stronger than IND-CCA. Given an IND-CCA encryption scheme $\mathcal{AE}$, we can easily produce a new IND-CCA encryption scheme $\mathcal{AE}' = (\mathcal{KG}', \mathcal{E}', \mathcal{D}')$ which is not N-PAT-IND-CCA:

$$\begin{array}{lll}
\mathcal{KG}'(\eta) = \mathcal{KG}(\eta) & & \\
\mathcal{E}'(m, pk) = m & & \text{if } m = sk \\
\qquad\qquad = \mathcal{E}(m, pk) & & \text{otherwise} \\
\mathcal{D}'(m, sk) = m & & \text{if } m = sk \\
\qquad\qquad = \mathcal{D}(m, sk) & & \text{otherwise}
\end{array}$$

This new scheme is obviously IND-CCA since $\mathcal{AE}$ is IND-CCA but without the restriction for N-PAT-IND-CCA, $\mathcal{AE}'$ is not N-PAT-IND-CCA.

Hence when a left-right pattern encryption oracle $\mathcal{E}^{Pat}_{pk_i}(LR(.,.,b))$ is given two patterns terms pat_0 and pat_1, it tests that none contains a pattern $[j]$ with $j < i$. If this happens, it outputs an error message, else it produces the encryption of the message corresponding to $pat_b : v(pat_b, ...)$ encoded by pk_i. This restriction is also justified by the existence of problems when encoding a key with itself for the DES algorithm. References concerning this restriction appear in [2].

The related criterion is γ_N where $\overline{c}$ is a list containing N pairs of keys (pk_i, sk_i). Oracles in $\overline{f}$ are the encryption oracles. They behave like the oracles defined in the previous example except that they perform the replacement of pattern variables with key values. The v operation is performed in the α layer whereas the β layer corresponds to the previous oracles (i.e. simple encoding). Formally, $f_i^{\alpha}(x) = v(x, \overline{c})$ and $f_i^{\beta}(x) = \mathcal{E}(x, sk_i)$. Oracles in $\overline{g}$ decrypt a message using secret keys as long as their argument has not been produced by an oracle in $\overline{f}$.

An asymmetric encryption scheme $\mathcal{AE}$ is said to be N-PAT-IND-CCA iff for any adversary $\mathcal{A}$, $\mathbf{Adv}^{\gamma_N}_{\mathcal{AE},\mathcal{A}}(\eta)$ is negligible. Note that 1-PAT-IND-CCA corresponds to IND-CCA.

Lemma 4.2 *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is N-PAT-IND-CCA, then $\mathcal{AE}$ is also IND-CCA.*

Proof: Let $\mathcal{A}$ be an IND-CCA adversary. We can easily construct a N-PAT-IND-CCA adversary $\mathcal{B}$ such that : $|\mathbf{Adv}_{\mathcal{A}}^{\gamma_1}(\eta)| = |\mathbf{Adv}_{\mathcal{B}}^{\gamma_N}(\eta)|$. Hence if $\mathcal{AE}$ is N-PAT-IND-CCA, the N-PAT-IND-CCA advantage (related to γ_N) of $\mathcal{B}$ is negligible and so is the IND-CCA advantage of $\mathcal{A}$. ■

Proposition 4.2 *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is N-PAT-IND-CCA, then $\mathcal{AE}$ is also (N+1)-PAT-IND-CCA.*

Proof: We have $c_i = (pk_i, sk_i)$. Then let γ' and γ'' be the partitions obtained with $na' = nb' = nc' = 1$; f_1^β and g_1^β only need c_1; f_1^α only needs $c_{2..(N+1)}$, this would not hold if we release the acyclicity hypothesis. Finally, f_i and g_i with $i \geq 2$ only need $c_{2..(N+1)}$ and so this is a valid partition. Hence, criterion γ_{N+1} has a valid partition constituted by γ_1 and γ_N.

The reduction theorem applies and gives:

$$|\mathbf{Adv}_{\mathcal{A}}^{\gamma_{N+1}}(\eta)| \leq 2.|\mathbf{Adv}_{\mathcal{B}}^{\gamma_1}(\eta)| + |\mathbf{Adv}_{\mathcal{A}^o}^{\gamma_N}(\eta)|$$

By hypothesis, $\mathcal{AE}$ is N-PAT-IND-CCA (hence IND-CCA). Then advantages of $\mathcal{B}$ and $\mathcal{A}^o$ are negligible and we can conclude that the advantage of $\mathcal{A}$ is negligible too. ■

Corollary 4.2 *For any N, $\mathcal{AE}$ is IND-CCA if and only if $\mathcal{AE}$ is also N-PAT-IND-CCA.*

This result tells us that if an encryption scheme is IND-CCA secure, then it is still secure when adding the possibility to ask for encryption of patterns instead of just encryption of messages.

4.3 Signature

In order to extend previous results to the case of protocols using signature, we present here a new definition of security for signature scheme, UNF-CCA, which is an adaptation of Selective (Un)Forgery Against Adaptive Chosen Message Attack [13].

The main requirement is that an adversary should not be able to forge a pair containing a message m and the signature of m using the secret signature key. An N-UNF-CCA adversary $\mathcal{A}$ is given N verification keys and has to produce a message and its signature under one of the keys. It has access to the security parameter η, N verification keys vk_i and N signature oracles $\mathcal{S}_{sik_i}(.)$. The experiment outputs bit 1 if $\mathcal{A}$ managed to produce a compromising pair $(m, \{m\}_{sik_i})$ which right part is not the result of a call to a signature oracle. Otherwise, the experiment outputs bit 0. Formally the experiment is detailed below.

Experiment $\mathbf{Exp}_{\mathcal{SS},\mathcal{A}}^{N-UNF}(\eta)$:
  for $i = 1$ **to** N **do**
    $(sik_i, vk_i) \overset{R}{\leftarrow} \mathcal{KG}(\eta)$
  $(m, \sigma) \overset{R}{\leftarrow} \mathcal{A}/\eta, vk_1, ..., vk_N,$
      $\mathcal{S}_{sik_1}(.), ..., \mathcal{S}_{sik_N}(.),$
  if σ is a valid signature of m under one of the sik_i
      not produced by $\mathcal{S}_{sik_i}(.)$
    return 1
  else return 0

The advantage of adversary $\mathcal{A}$ in winning the UNF-CCA challenge is defined as:

$$\mathbf{Adv}_{\mathcal{SS},\mathcal{A}}^{N-UNF}(\eta) = Pr[\mathbf{Exp}_{\mathcal{SS},\mathcal{A}}^{N-UNF}(\eta) = 1]$$

A signature scheme $\mathcal{SS}$ is said to be N-UNF-CCA iff for any adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{SS},\mathcal{A}}^{N-UNF}(\eta)$ is negligible. Instead of 1-UNF-CCA, we write UNF-CCA.

As the challenge is not anymore guessing the value of a bit b, our reduction theorem cannot apply directly. However, by modifying the proof scheme given above, it is possible to deduce the following property relating the UNF-CCA and N-UNF-CCA criteria. The proof is given inappendix B .

Proposition 4.3 *For any signature scheme $\mathcal{SS}$, if $\mathcal{SS}$ is UNF-CCA, then it is also N-UNF-CCA.*

4.4 N-PAT-UNF-IND-CCA

To be able to deal with protocols using both an encryption scheme and a signature scheme, we define a new criterion N-PAT-UNF-CCA: a combination of N-PAT-IND-CCA and N-UNF-CCA. Let us consider an asymmetric encryption scheme $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ and a signature scheme $\mathcal{SS} = (\mathcal{KG}', \mathcal{S}, \mathcal{V})$. We use two types of adversary: those who try to find the secret bit b used in the N left-right pattern encryption oracles and those who try to produce a message and its signature under one of the N challenge signature keys. The left-right pattern encryption oracles accept patterns of the form $[sik_i]$ where sik_i is one of the challenge signature keys. Experiments $N - PUI_1$ and $N - PUI_2$ are defined in Figure 2 given in the appendix. Then, the corresponding advantages are:

$$\mathbf{Adv}^{N-PUI_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta) = Pr[\mathbf{Exp}^{N-PUI_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta, 1) = 1] - Pr[\mathbf{Exp}^{N-PU_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta, 0) = 1]$$

$$\mathbf{Adv}^{N-PUI_2}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta) = Pr[\mathbf{Exp}^{N-PUI_2}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta) = 1]$$

A couple $(\mathcal{AE}, \mathcal{SS})$ is said to be N-PAT-UNF-IND-CCA iff for all adversary $\mathcal{A}$, $\mathbf{Adv}^{N-PUI_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta)$ and $\mathbf{Adv}^{N-PUI_2}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta)$ are negligible.

The following property states that the combination of a secure signature scheme and a secure encryption scheme is still secure. Its proof can be done using the same proof scheme as for the reduction theorem.

Proposition 4.4 *If $\mathcal{AE}$ is N-PAT-IND-CCA and if $\mathcal{SS}$ is N-UNF-CCA, $(\mathcal{AE}, \mathcal{SS})$ is N-PAT-UNF-IND-CCA.*

To sum up, we proved the following equivalences between criteria:

$$\begin{aligned} \text{IND} - \text{CCA} &\Leftrightarrow \text{N} - \text{PAT} - \text{IND} - \text{CCA} \\ \text{UNF} - \text{CCA} &\Leftrightarrow \text{N} - \text{UNF} - \text{CCA} \\ (\text{IND} - \text{CCA}, \text{UNF} - \text{CCA}) &\Leftrightarrow \text{N} - \text{PAT} - \text{UNF} - \text{IND} - \text{CCA} \end{aligned}$$

5 Dolev-Yao is a Safe Abstraction

In this section, we give a precise formalization of the link between the two commonly used approaches for verification of cryptographic protocols, i.e. the computational approach and the formal approach. For that purpose, we first define cryptographic protocols, then we relate traces from both models. This relation is used to prove the main theorem.

5.1 Description of Cryptographic Protocols

A multi-party protocol is defined by a list of triples (m_1, m_2, R), called actions. The action (m_1, m_2, R) means that an agent playing role R sends a message m_2 after receiving a message m_1. It is possible to replace m_1 with an empty message denoted by "-" for the first action of the protocol. For the last action, the same thing can be done with m_2. A role R represents a program that is executed by an agent Ag during a session S. In session S, we say that agent Ag impersonates role R. Let us consider the Needham-Schroeder-Lowe protocol (NSL introduced in [17]) there are two roles: the initiator and the receiver.

For a session S, an agent Ag has some local variables used during the execution of his role: his local copy of variable Var is denoted by $Ag.S.Var$. Each agent has five variables for each of his sessions: $Init, Rec, Na, Nb, Pc$. The list of actions is:

- $(-, \{\langle Init, Na\rangle\}_{Pk_{Rec}}, Init)$
- $(\{\langle init, na\rangle\}_{Pk_{Rec}}, \{\langle Rec, na, Nb\rangle\}_{Pk_{init}}, Rec)$
- $(\{\langle Rec, Na, nb\rangle\}_{Pk_{Init}}, (\{nb\}_{Pk_{Rec}}, Init)$
- $((\{Nb\}_{Pk_{Rec}}, -, Rec)$

We make a distinction between values known before an action and values received during the action: values already known are denoted using a capital for their first letter. Let Ag be an agent impersonating the initiator. In the first action, Ag chooses B as a receiver, and the value of nonce Na for session s. He sets variables $Ag.s.Init$ to Ag, $Ag.s.Rec$ to B and $Ag.s.Na$ with the chosen value. Then he sends message $\{\langle Ag.s.Init, Ag.s.Na\rangle\}_{Ag.s.Rec}$. In the second action, B receives a message encrypted with his public key. He decrypts it and parses the plain-text as a pair $\langle init, na\rangle$. After that, he chooses a new session number s' and sets $B.s'.Init$ to $init$, $B.s'.Rec$ to B, $B.s'.Na$ to na and finally chooses a fresh value for $B.s'.Nb$. Finally, he sends message $\{\langle B.s'.Rec, B.s'.Na, B.s'.Nb\rangle\}_{Pk_{B.s'.Init}}$ to Ag. The remaining actions have similar semantics.

Hypothesis over Protocols

The following restrictions are made over the protocol Π considered in this section. Π has to be executable, that is each role can be run by a PRTM. In the formal world, for any execution, secret keys of honest agents remain secrets, there is no encryption cycle, there is a nonce in each signed message that also remains secret. Moreover, we ask that messages include enough typing information to allow parsing of received messages. We also assume that any agent knows identities of all the other agents.

Computational and Formal Models

For both models, agents involved in a protocol send their messages through a network. This network is modeled by the Adversary. The Adversary intercepts any message sent by an agent. He can forge new messages using his knowledge and send these messages to agents usurping an agent's identity.

In the formal model, the Adversary is a classical Dolev-Yao Intruder [11]. In the computational model, both the Adversary and the implementation of the protocol are PRTM, denoted by $\mathcal{A}_c$ and Π_c. Π_c is used as an oracle by $\mathcal{A}_c$. Messages are bit-strings, new nonces generated by agents are random bit-strings. Keys used by agents are generated using $\mathcal{KG}$. Encryptions and decryptions are obtained using algorithms from $\mathcal{AE}$. Signatures related functions use $\mathcal{SS}$. R_Π denotes random coins used by Π_c, for generation of keys, of nonces and any other random action used during an execution of the protocol. $R_\mathcal{A}$ denotes the random coins used by an Adversary $\mathcal{A}$. The implementation of the protocol (i.e. behavior of the different agents) is given to the adversary through an oracle. If R_Π and $R_\mathcal{A}$ are fixed, the behavior of $\mathcal{A}_c$ against the oracle Π_c is deterministic.

$\mathcal{A}_c$ can create new valid identities and thus impersonate some dishonest agents.

5.2 Non Dolev-Yao Traces

The main result of this section is that if the encryption scheme used in the implementation of Π is IND-CCA, then the computational Adversary acts as a formal adversary with overwhelming probability. A trace is a list of tuples (m_1, m_2, Ag, s) called transitions where m_1 is a message sent by the Adversary to agent Ag for session s and m_2 is the answer from Ag. As before, message m_1 and m_2 can be "-". Assignment $t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c$ denotes that the trace t_c is obtained by the computational Adversary $\mathcal{A}_c$ (using the random coins $R_\mathcal{A}$)confronted to Π_c (using the random coins R_Π). We assume that only messages accepted by an agent appear in the trace. We now transform a computational trace into a *pseudo formal trace*. The resulting trace is only pseudo formal because even if messages are expressed using Dolev-Yao terms, this does not imply that there exists a formal Adversary producing this trace. The transformation given here can be seen as verification of the trace by all the honest agents working together. Their goal is to check if the adversary performed an action that is not in the Dolev-Yao model. To achieve this, messages in the computational trace (which are bit-strings) are replaced with Dolev-Yao terms using the following:

- Bit-strings corresponding to identities are associated to fresh atoms: $H_1, H_2, ...$ for honest agents and $I_1, I_2, ...$ for dishonest agents.
- Bit-strings corresponding to long-term keys are associated to fresh keys: $Pk_{H_1}, Pk_{H_2}, ...$ and $Pk_{I_1}, Pk_{I_2}, ...$ for public keys, Sk_{Id} for associated secret keys.

- Bit-strings corresponding to nonces N generated in session S by an honest agent are associated to fresh atoms N^s.
- Bit-strings corresponding to fresh public or secret keys generated by an honest agent in session s are associated with fresh keys Pk^s and Sk^s.
- Bit-strings corresponding to keys generated by the Adversary are associated with fresh keys Pk_I^j and Sk_I^j.
- Bit-strings corresponding to concatenation of a message associated to term T_1 with a message associated to term T_2 are associated with $\langle T_1, T_2 \rangle$.
- Bit-strings corresponding to encryption of a message associated to term T with a key associated to term K are associated to term $\{T\}_K$.

Note that this is not complete as we have not yet taken into account nonces generated by the Adversary. As the Adversary is a PRTM, whenever he has to send a new nonce, he does not have to generate it randomly: he can send composed messages instead of nonces or perform operations over bit-strings (XOR, changing bit order, adding or removing bits...). Hence for the honest agents it is impossible to guess how the Adversary has chosen his nonces. This is why when transforming a bit-string corresponding to a message where an honest agent receives a new nonce, we only test if the corresponding bit-string is an already known nonce. In this case, the bit-string is associated to the nonce term. Else, it is associated to a fresh variable X_i. If later this bit-string is parsed as something else (tuple, encoding), variable X_i is replaced by the appropriate term. The same thing is done when an honest agent receives a message encrypted with a key which inverse is not known or a signature impossible to verify (at reception time). When every message in the trace has been transformed, each remaining X_j is replaced by the fresh atom N_I^j, i.e. remaining variables are considered as fresh nonces. The pseudo-formal trace corresponding to computational trace t_c is denoted by $\alpha(t_c)$. Two computational traces are said *equivalent* if they have the same pseudo formal trace.

Definition 5.1 (Non Dolev-Yao Traces) *A formal trace t_f is said* Non Dolev-Yao *(NDY) iff there exists a message sent by the Adversary which cannot be deduced from previous messages using Dolev-Yao's deduction, this message is called a NDY message. A computational trace t_c is said NDY iff $\alpha(t_c)$ is NDY.*

5.3 A Computational Trace is Certainly a Dolev-Yao Trace

In this section, we prove that if the encryption and signature schemes verify IND-CCA resp. UNF-CCA and if the number of possible nonces is exponential in η, then the probability that a computational trace is NDY is negligible. This means that the computational Adversary, even with all the computing power of PRTM, cannot have a behavior not represented by a formal adversary. We first state this with only encryption then extend it to the case of encryption and signature.

Theorem 5.1 *Let Π be a protocol. Let $\mathcal{AE}$ be the encryption scheme used in Π_c. If $\mathcal{AE}$ is IND-CCA then for any concrete Adversary $\mathcal{A}_c$:*
$Pr\left[t_c \overset{R_{\mathcal{A}}, R_\Pi}{\leftarrow} \mathcal{A}_c/\eta, \Pi_c \ ; \ t_c \ NDY\right]$ is negligible.

Proof: As the proof of this theorem is complex, it is presented in appendix D.

Here, we use a simple example to show how the proof works. Let us consider that $\mathcal{A}_c/\eta, \Pi_c$ produces a trace t_c such that

$$\alpha(t_c) = (-, \{A, N\}_{pk}, A, s); (\{C, N\}_{pk}, -, A, s)$$

As key sk is secret, t_c is non Dolev-Yao: message $\{C, N\}_{pk}$ is not deducible because N is not deducible. Hence, it is possible to build an Adversary $\mathcal{B}$ against IND-CCA by using nonce N.

For that purpose, $\mathcal{B}$ generates randomly an identity A and two different nonces N_0 and N_1. $\mathcal{B}$ then calls its encryption oracle with parameters $\langle A, N_0 \rangle$ and $\langle A, N_1 \rangle$ which gives $\{A, N_b\}_{pk}$ (b is the challenge bit of IND-CCA). $\mathcal{B}$ then executes $\mathcal{A}_c$ and gives it the previous message when $\mathcal{A}_c$ asks for an input. $\mathcal{A}_c$ outputs $\{C, N_b\}_{pk}$

that $\mathcal{B}$ immediately submits to its decryption oracle. Hence, $\mathcal{B}$ obtains N_b, compares it to N_0 and N_1 and so knows b. At this stage, $\mathcal{B}$ is able to win the IND-CCA challenge.

By applying this method, it is possible to gain the challenge against IND-CCA for any non Dolev-Yao trace. Thus, the probability of producing a non Dolev-Yao trace is negligible. ■

This theorem can be extended to protocols using both an encryption scheme and a signature scheme.

Theorem 5.2 *Let Π be a protocol. Let $\mathcal{AE}$ be the encryption scheme and $\mathcal{SS}$ the signature scheme used in Π_c. If $\mathcal{AE}$ is IND-CCA and $\mathcal{SS}$ is UNF-CCA then for any concrete Adversary $\mathcal{A}_c$:*
$Pr\left[t_c \overset{R_{\mathcal{A}}, R_\Pi}{\leftarrow} \mathcal{A}_c/\eta, \Pi_c \ ; \ t_c \ NDY\right]$ *is negligible.*

Proof: The proof of this theorem is presented in appendix E. ■

5.4 Formal and Computational Properties

Let P_c be a property in the computational world represented by a predicate over computational traces. A protocol Π verifies P_c (denoted by $\Pi \models_c P_c$) iff for any Adversary $\mathcal{A}_c$,
$Pr\left[t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c \ ; \ \neg P_c(t_c)\right]$ is negligible. A property in the formal world is represented by a predicate P_f over formal traces. Hence, a protocol Π verifies P_f (denoted by $\Pi \models_f P_f$) iff any trace produced by a Dolev-Yao adversary against Π verifies P_f.

Using theorem 5.2, we prove the following result which states that proving formally P_f allows us to deduce P_c.

Theorem 5.3 *Let P_f and P_c be a formal and a computational property such that*

$$\forall t_c, \forall t_f, \left(P_f(t_f) \wedge \alpha(t_c) = t_f\right) \Rightarrow P_c(t_c)$$

If $\mathcal{AE}$ is IND-CCA and $\mathcal{SS}$ is UNF-CCA, then

$$\Pi \models_f P_f \Rightarrow \Pi \models_c P_c$$

This theorem states that if the formal property correctly under-approximates the computational property then the formal abstraction is correct.

This theorem has been applied to mutual authentification in [20] and holds for nonce secrecy [10].

6 Conclusion

The main result of this paper is a significant step towards reconciling the computational and the formal approaches to security. It is a contribution to a research endeavor popularized, maybe even initiated, by Abadi and Rogaway [2] and pursued by, e.g., Micciancio and Warinschi [25, 20], Herzog [15] and others. It is the problem of relating the formal approach which assumes perfect cryptographic primitives on one hand and the more detailed computational approach on the other hand. This question comes in two different forms: 1.) passive intruder is considered and 2.) active intruder that interacts with participants running a protocol. The work of Abadi and Rogaway falls in the first category. Micciancio and Warinschi [20] work belongs to the second category. They proved that under some restrictions on the protocol, if the encryption scheme is IND-CCA then the formal model is a safe abstraction of the computational. The restrictions they put are, however, too restrictive. In this paper, we considered active intruders. Our main result is that an adversary behavior follows the formal model with overwhelming probability, if the encryption scheme is IND-CCA and the signature scheme is UNF-CCA. This result has immediate applications as automatic verification of security protocols is quite developed now and as there are encryption algorithms that verify the required properties. Our result extends previous ones and allow:

- Multi-party protocols.
- More cryptographic primitives: combination of digital signature and asymmetric encryption.

- Protocols where encoding of secret keys and message forwarding are allowed.

A second main contribution of our paper is a formal definition for security criteria and a reduction theorem. This theorem and its proof scheme seem to apply in a wide variety of cases. It allows to prove equivalences between a security criterion and some of its sub-criteria. This theorem allowed us to give a quick proof of already known results, to generalize this to new results and we believe that it could be useful whenever one wants to relate two security criteria.

Concerning extensions of this work, in [21], we extend these results to protocols using simultaneously all the classical cryptographic primitives: asymmetric and symmetric encoding, signature and hashing. This paper also deals with simple equational theories.

When considering the famous Clark and Jacobs survey [8], previous results can only apply to the Needham-Schroeder-Lowe protocol, this can now be applied to a wider class of protocols as shown in appendix A. In fact, our generalisation (including results from this paper and [21]) allows us to deduce security in the computational model for all the protocols in this survey that are secure in the formal model.

Acknowledgments

This work has been partially supported by the RNTL project PROUVE-03V360 and the ACI project ROSSIGNOL

References

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *31th International Colloquium on Automata, Languages and Programming (ICALP'04)*, 2004. 1

[2] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany. 1, 4.2, 6

[3] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Proc. CRYPTO 94*, pages 341–358. Springer, 1994. Lecture Notes in Computer Science No. 839. 1

[4] B. Blanchet. Abstracting cryptographic protocols by prolog rules. In *International Static Analysis Symposium*, volume 2126 of *LNCS*, pages 433–436, 2001. 1

[5] A. Boldyreva, M. Bellare, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology-EUROCRYPT 2000, Lecture Notes in Comput. Sci., Vol. 1807*, pages 259–274. Springer, 2000. 3.1, 4.1

[6] L. Bozga. *Methods and Tools for Verification of Security Protocols*. PhD thesis, Université Joseph Fourier - VERIMAG, 2004. A

[7] L. Bozga, Y. Lakhnech, and M. Périn. Hermes: An automatic tool for verification of secrecy in security protocols. In *15th International Conference on Computer Aided Verification (CAV),*, volume 2725 of *LNCS*, 2003. 1, A

[8] John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Version 1.0, University of York, Department of Computer Science, November 1997. 1, 6

[9] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the eighteenth annual IEEE symposium on Logic In Computer Science*. IEEE Computer Society Press, June 2003. 1

[10] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. Research Report RR-5341, INRIA, October 2004. 5.4

[11] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. 1, 2.1, 2.2, 5.1

[12] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984. 1

[13] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 4.3

[14] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, volume 1800 of *LNCS*, 2000. 1

[15] J. Herzog. *Computational soundness for standard assumptions of formal cryptography*. PhD thesis, MIT, 2002. 6

[16] P. Laud. Symmetric encryption in automatic analyses for confidentiality against adaptive adversaries. In *Proc. of 2004 IEEE Symposium on Security and Privacy*, pages 71–85, 2004. 1

[17] G. Lowe. An attack on the Needham-Schroeder public-key authentification protocol. *Information Processing Letters*, 56(3):131–133, 1995. 1, 5.1

[18] Gavin Lowe. Analysing protocols subject to guessing attacks. In *Proc. of the Workshop on Issues in the Theory of Security (WITS'02)*, 2002. 1

[19] B. Pfitzmann M. Backes and M. Waidner. U universally composable cryptographic library. In ACM Press, editor, *Computer and Communication Security*, Oct. 2003. 1

[20] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings of the Theory of Cryptography Conference*, pages 133–151. Springer, 2004. 1, 3, 5.4, 6, A

[21] Y. Lakhnech R. Janvier and L. Mazaré. (de)compositions of cryptographic schemes and their applications to protocols. Technical Report TR-2005-03, Verimag, Centre Équation, 38610 Gières, January 2005. (document), 4.1, 6

[22] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *IEEE Computer Security Foundations Workshop*, 2001. 1, 2.1

[23] D. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW '99)*, pages 192–202, Washington - Brussels - Tokyo, June 1999. IEEE. 1

[24] http://www-eva.imag.fr/. 1

[25] B. Warinschi. A computational analysis of the needham-schroeder(-lowe) protocol. In *Proceedings of 16th Computer Science Foundation Workshop*, pages 248–262. ACM Press, 2003. 1, 6

A Appendix: Security Results

The following table summarizes the protocols on which we can use our result to prove secrecy in the computational model. Secrecy in the Dolev-Yao model has been checked using HERMES (see [7]), a protocol verifier capable of dealing with an infinite number of sessions. In the HERMES results column, OK means that the protocol has been successfully verified for the secrecy property and so the protocol is secure in Dolev-Yao model (as the abstraction made by HERMES to deal with an infinite number of sessions is safe). For more details on HERMES verification of these protocols, see [6].

Then, for protocols that are secure in the formal model, the Computational results tells whether the protocol fits in our framework or not and so OK means that the protocol is safe in the computational model. The letters in parenthesis indicate which extension of the previous results (in particular, [20] where any exchanged messages has to be an encryption of nonces and identities using the public key of the next agent) is useful: s means symmetric encryption, k encryption of keys under keys, l encryption inside an encrypted message and n nonces without encoding.

Protocol Name	Hermes (Formal)	Computational
Yahalom	OK	OK(sk)
Needham-Schroeder Public Key	Attack	-
Needham-Schroeder-Lowe	OK	OK
Otway-Rees	OK*	OK(sk)
Denning Sacco Key Distribution with Public Key	Attack	-
Wide Mouthed Frog (modified)	OK	OK(skn)
Kao-Chow	OK	OK(skn)
Neumann-Stubblebine	OK*	OK(skn)
Needham-Schroeder Symmetric Key	Attack	-
TMN	Attack	-
ISO-Symmetric-Key-One-Pass-Unilat.-Auth.	OK	OK(skn)
ISO-Symmetric-Key-Two-Pass-Unilat.-Authent.	OK	OK(sn)
Andrew Secure RPC	OK	OK(skl)
Woo and Lam Mutual Authentication (modified)	OK	OK(skn)
Skeme (modified)	OK	OK(skn)

* There is a known attack of the untyped version of the protocol. Discovering this type attack automatically requires to deal with non-atomic keys. This is not yet implemented in HERMES.

B Proof of the Reduction Theorem

This section provides details of the proof of the reduction theorem. Let $\mathcal{A}$ be an adversary to the criterion γ. The machines $\mathcal{A}^0$ and $\mathcal{B}$ use $\mathcal{A}$ as a sub-process and since they do not have access to all the oracles required by $\mathcal{A}$, they simulate the missing oracles. The adversary $\mathcal{A}^o$ needs to generate $c_{1..na'}$ and to construct the associated oracles from $\overline{f}$ and $\overline{g}$. As it is impossible to build the correct oracles, instead we use fake oracles $g_{1..nc'}$ using the generated parameters. The fake parameters are denoted by ${}_{fake}c_i$, the corresponding fake oracles ${}_{fake}f_i$ and ${}_{fake}g_i$. As the partition is valid, all the arguments needed by fake oracles are included in fake parameters and so $\mathcal{A}^o$ can be constructed as follows.

Adversary $\mathcal{A}^o$:

$_{fake}c_{1..na'} \xleftarrow{R} \Theta(\eta)$
$_{fake}f_i \leftarrow \lambda(l,r).f_i^\beta(f_i^\alpha(l, c^o_{na'+1..na}), {}_{fake}c_{1..na'})$ for i in $1...nb'$
$_{fake}g_i \leftarrow \lambda(x).g_i(x, {}_{fake}c_{1...na'})$ for i in $1...nc'$
$d \xleftarrow{R} \mathcal{A}/\eta, {}_{fake}f_{1..nb'}, {}_{fake}g_{1..nc'}$
return d

Fake oracles behave like normal oracles using generated parameters $_{fake}c_{1..na'}$: applied to x, $_{fake}g_i$ returns $g_i(x, {}_{fake}c_{1..na'})$ (for $i \leq nc'$). For $i \leq nb'$, $_{fake}f_i$ applied to (l,r) returns:

$$f_i^\beta(f_i^\alpha(l, c^o_{na'+1..na}), {}_{fake}c_{1..na'})$$

The left argument l is always chosen. $c^o_{na'+1..na}$ represents a known value for parameters $c_{na'+1}$ to c_{na} (for example, parameters obtained using tail for any random coin needed in their computation). The main characteristic of these parameters is to be fixed (which is useful as machine $\mathcal{B}$ also uses them).

Oracles given by $\mathcal{A}^o$ to $\mathcal{A}$ does not have the behavior expected by $\mathcal{A}$. Either this does not affect the advantage of $\mathcal{A}$, which means that $\mathcal{A}^o$ can use this against γ'', or $\mathcal{A}$ deviates from its normal behavior, which means that $\mathcal{A}$ obtains information over the challenge bit b in the left-right oracles faked in $\mathcal{A}^o$. In this case, the adversary $\mathcal{B}$ tries to distinguish between the normal behavior of $\mathcal{A}$, given a criterion γ, and the behavior of $\mathcal{A}$ with the oracles given as in $\mathcal{A}^o$. The adversary $\mathcal{B}$ picks a random bit b which will be the challenge bit given to $\mathcal{A}$. It also produces the missing (fake) parameters and will use the corresponding fake oracles. For oracles $_{fake}f_i$ when $i > nb'$, these oracles can be faked using bit b. Else, for $i \leq nb'$, it uses its own challenge bit b' and when called using m_0, m_1, it returns the result of oracle f_i^β called with parameters $f_i^\alpha(m_0, c^o_{na'+1,..na})$ and $f_i^\alpha(m_b, {}_{fake}c_{na'+1,..na})$. The parameter is chosen according to the value of bit b' of criterion γ'. If $b' = 0$, oracles given to $\mathcal{A}$ are exactly the oracles given to $\mathcal{A}^o$ and if $b' = 1$, oracles correspond to normal γ oracles. To solve its own challenge (i.e. return the value of b'), $\mathcal{B}$ assumes that $\mathcal{A}$ has a better advantage if it is given the normal oracles. Hence if $\mathcal{A}$ wins its challenge by answering b, $\mathcal{B}$ answers the bit 1. Otherwise it answers 0.

Adversary $\mathcal{B}$:

$b \xleftarrow{R} [0,1]$
$_{fake}c_{na'+1..na} \xleftarrow{R} \Theta(\eta)$
if $b = 0$ **then**
  $_{fake}f_i \leftarrow \lambda(l,r).\underline{f_i^\beta}(f_i^\alpha(l, c^o_{na'+1,..na}), f_i^\alpha(l, {}_{fake}c_{na'+1,..na}))$ for i in $1...nb'$
  $_{fake}f_i \leftarrow \lambda(l,r).\overline{f_i^\beta}(f_i^\alpha(l, {}_{fake}c_{na'+1,..na}), {}_{fake}c_{na'+1,..na})$ for i in $nb'+1...nb$
else
  $_{fake}f_i \leftarrow \lambda(l,r).\underline{f_i^\beta}(f_i^\alpha(l, c^o_{na'+1,..na}), f_i^\alpha(r, {}_{fake}c_{na'+1,..na}))$ for i in $1...nb'$
  $_{fake}f_i \leftarrow \lambda(l,r).\overline{f_i^\beta}(f_i^\alpha(r, {}_{fake}c_{na'+1,..na}), {}_{fake}c_{na'+1,..na})$ for i in $nb'+1...nb$
$_{fake}g_i \leftarrow \lambda(x).g_i(x, {}_{fake}c_{na'+1...na})$ for i in $nc'+1...nc$
$d \xleftarrow{R} \mathcal{A}/\eta, {}_{fake}f_{1..nb}, {}_{fake}g_{nc'+1..nc}$
if $d = b$ **return** 1
else return 0

When $b' = 1$, $\mathcal{B}$ outputs 1 whenever $\mathcal{A}$ wins its challenge. Hence:

$$\begin{aligned} 2 \times Pr[\mathbf{Exp}^{\gamma'}_{\mathcal{B}}(\eta,1) = 1] &= Pr[\mathbf{Exp}^{\gamma'}_{\mathcal{B}}(\eta,1) = 1 | b = 1] + Pr[\mathbf{Exp}^{\gamma'}_{\mathcal{B}}(\eta,1) = 1 | b = 0] \\ &= Pr[\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta,1) = 1] + Pr[\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta,0) = 0] \\ &= 1 + \mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) \end{aligned}$$

A similar calculus gives that $2 \times Pr[\mathbf{Exp}^{\gamma'}_{\mathcal{B}}(\eta,0) = 1] = 1 + \mathbf{Adv}^{\gamma''}_{\mathcal{A}^o}(\eta)$.

Thus, $2 \times \mathbf{Adv}^{\gamma'}_{\mathcal{B}}(\eta) = \mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) - \mathbf{Adv}^{\gamma''}_{\mathcal{A}^o}(\eta)$. This gives, with the triangle inequality:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma'}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma''}_{\mathcal{A}^o}(\eta)|$$

C Signature Result

Let $\mathcal{SS}$ be a UNF-CCA signature scheme. Let $\mathcal{A}$ be a polynomial N-UNF-CCA adversary. We give the construction of an UNF-CCA adversary $\mathcal{B}$ such that $\left|N.\mathbf{Adv}^{N-UNF}_{\mathcal{SS},\mathcal{A}}(\eta)\right| \leq \left|\mathbf{Adv}^{UNF}_{\mathcal{SS},\mathcal{B}}(\eta)\right|$. The idea is that there is no common knowledge shared between the N signature oracles given to $\mathcal{A}$, unlike the bit b shared by the left-right oracles in the case of N-IND-CCA. Moreover, to win the challenge, $\mathcal{A}$ only has to produce a signature under one of the signature keys sik_i. The adversary $\mathcal{B}$ is given a challenge $(\eta, vk, \mathcal{S}_{sik}(.), \mathcal{V}_{vk}(.,.))$. It can run $\mathcal{A}$ as a sub-process giving him vk, $\mathcal{S}_{sik}(.)$ and $\mathcal{V}_{vk}(.,.)$ instead of oracles related to (sik_i, vk_i): vk_i, $\mathcal{S}_{sik_i}(.)$ and $\mathcal{V}_{vk_i}(.,.)$. $\mathcal{B}$ also generates the $N-1$ other pairs of keys and simulates the associated oracles. Eventually, $\mathcal{B}$ outputs the pair (m,σ) produced by $\mathcal{A}$. Since $\mathcal{B}$ does not know *a priori* which key will be falsified, it first picks a random number i in $[1,..,N]$ and bets that $\mathcal{A}$ will try to falsify sik_i. Formally we define $\mathcal{B}$ as:

Adversary $\mathcal{B}$:
$i \xleftarrow{R} [1,..,N]$
for $j = 1$ **to** N **do**
  if $i = j$
    $vk_i \leftarrow \underline{vk}$
    $\mathcal{S}_{sik_i} \leftarrow \lambda x.\underline{\mathcal{S}_{sik}}(x)$
    $\mathcal{V}_{vk_i} \leftarrow \lambda(m,\sigma).\underline{\mathcal{V}_{vk}}(m,\sigma)$
  else
    $(sik_j, vk_j) \xleftarrow{R} \mathcal{KG}(\eta)$
    construct $\mathcal{S}_{sik_j}(.)$ and $\mathcal{V}_{vk_j}(.,.)$
$(m,\sigma) \xleftarrow{R} \mathcal{A}/\eta, vk_1, ..., vk_N,$
    $\mathcal{S}_{sik_1}(.), ..., \mathcal{S}_{sik_N}(.),$
    $\mathcal{V}_{vk_1}(.,.), ..., \mathcal{V}_{vk_N}(.,.)$
return (m,σ)

The adversary $\mathcal{B}$ wins its challenge whenever it has chosen the right key and $\mathcal{A}$ outputs a valid signature under this key. We then obtain:

$$\left|N.\mathbf{Adv}^{N-UNF}_{\mathcal{SS},\mathcal{A}}(\eta)\right| \leq \left|\mathbf{Adv}^{UNF}_{\mathcal{SS},\mathcal{B}}(\eta)\right|$$

We should also note that the adversary $\mathcal{B}$ is polynomially bounded iff $\mathcal{A}$ is polynomially bounded too. Hence if $\mathcal{SS}$ is UNF-CCA, the UNF-CCA advantage of $\mathcal{B}$ is negligible and so is the N-UNF-CCA advantage of $\mathcal{A}$.

D Proof of Theorem 2

The intuition is that if $\mathcal{A}_c$ produce a NDY trace, then he had to break the encryption scheme $\mathcal{AE}$. Let Q be the polynomial bounding the execution of $\mathcal{A}_c$. We build a Q-PAT-IND-CCA Adversary $\mathcal{B}$ such that

$$Pr\left[t_c \xleftarrow{R} \mathcal{A}_c/\eta, \Pi_c \ ; \ t_c \ NDY\right] \leq Q(\eta)^2.n_N.\mathbf{Adv}^{Q-PAT}_{\mathcal{AE},\mathcal{B}}(\eta) \tag{1}$$

where n_N is a the number of nonces used in the protocol description. If $\mathcal{AE}$ is IND-CCA, the Q-PAT-IND-CCA advantage of $\mathcal{B}$ is negligible (Corollary 3.2). Hence, the probability for $\mathcal{A}_c$ to produce a non Dolev-Yao trace is negligible.

The Q-PAT-IND-CCA Adversary $\mathcal{B}$ uses $\mathcal{A}_c$ as a subroutine and will deduce the challenge bit b as soon as $\mathcal{A}_c/\eta, \Pi_c$ produces a non Dolev-Yao trace. Using his own oracles, $\mathcal{B}$ simulates the oracle Π_c (used by $\mathcal{A}_c$) and produces the pseudo formal trace in order to find NDY messages.

First, let us consider the case where all the non Dolev-Yao traces produced by $\mathcal{A}_c/\eta, \Pi_c$ are equivalent (i.e. have the same associated pseudo-formal trace). Let m_{NDY} be the first NDY message and N be the nonce (generated by agent H in session s) that makes this message NDY. In order to answer queries from $\mathcal{A}_c$, $\mathcal{B}$ randomly generates identities for honest agents, nonces created by honest agents except N. $\mathcal{B}$ uses his challenge keys for their public keys. For the nonce $H.s.N$, $\mathcal{B}$ generates two nonces N_0 and N_1.

When $\mathcal{A}_c$ waits for a message m, $\mathcal{B}$ has to forge the message: if the message does not contain $H.s.N$, it can be computed without using his left-right encryption oracles, else as m_{NDY} is non Dolev-Yao, $H.s.N$ is encoded by a public key in m. Let $\{m'\}_{pk}$ be the biggest sub-message of m which is an encryption. $\mathcal{B}$ can compute m'_0 and m'_1 (m'_b is equal to m' where N_b replaces $H.s.N$) and uses the left-right encryption oracle pk on (m'_0, m'_1) to build m.

When $\mathcal{A}_c$ emits a message m, m is parsed according to the protocol specification (as described previously for pseudo formal trace). During parsing, if $\mathcal{B}$ has to decrypt a message then either this message has been produced using a left-right encryption oracle and there is no new information inside or $\mathcal{B}$ can use his decryption oracles ($\mathcal{B}$ only has to decrypt messages encrypted using keys which inverse is known by an honest agent). Eventually, when $\mathcal{B}$ receives m (in the case of a NDY trace), he uses his decryption oracles to obtain N_b. Then, $\mathcal{B}$ knows the challenge bit b and wins its challenge. Note that nonce N_b cannot be protected under an encoding produced by an encryption oracle: then message m would not be non Dolev-Yao because of $H.s.N$.

The probability that $\mathcal{B}$ obtains N_b is equal to the probability that $\mathcal{A}_c$ produces a NDY trace. Since $\mathcal{B}$ does not know *a priori* which nonce $H.s.N$ to choose, he randomly picks a nonce N among the n_N nonces appearing in the description of the protocol, an honest agent H among $Q(\eta)$ possible, a session s among $Q(\eta)$ possible sessions.

$$Pr\left[\mathcal{B}\ obtains\ N_b\right] \geq \frac{Pr\left[t_c \overset{R_{\mathcal{A}}, R_\Pi}{\leftarrow} \mathcal{A}_c/\eta, \Pi_c\ ;\ t_c\ NDY\right]}{Q(\eta).Q(\eta).n_N}$$

This is not an equality since a trace can be NDY because of two nonces. However, $\mathcal{B}$ may not be able to build a message containing $H.s.N$ (as this nonce can occur not protected by a public key associated to an oracle, $\mathcal{B}$ does not know which value to choose N_0 or N_1). In this case and when $\mathcal{B}$ does not obtain N_b, he answers a random bit. Else, he answers b. Note that as $\mathcal{A}_c$ is a PRTM, $\mathcal{B}$ is a PRTM too. The advantage of $\mathcal{B}$ is:

$$Pr[\mathbf{Exp}_{\mathcal{B}}^{Q-PAT}(\eta, 1) = 1] = Pr[\mathcal{B}\ obtains\ N_b | b = 1] + \frac{Pr[\mathcal{B}\ does\ not\ obtain\ N_b | b = 1]}{2}$$

$$Pr[\mathbf{Exp}_{\mathcal{B}}^{Q-PAT}(\eta, 0) = 1] = \frac{Pr[\mathcal{B}\ does\ not\ obtain\ N_b | b = 0]}{2}$$

The event "$\mathcal{B}$ does not obtain N_b" is independent from the value of b and so formula 1 holds.

[**Adding Secret Keys**] We can now extend the proof for protocols in which fresh keys are generated. Fresh keys are more difficult to handle because, unlike long term keys, a fresh secret key can be shared with a dishonest agent, even if it was generated by an honest agent. Hence, when an honest agent has to generate a fresh key, two distinct cases may occur. In the first case, the secret key will only be shared with honest agents. As it will only appear in messages encrypted with the public key of an honest agent, $\mathcal{B}$ can use one of his challenge key and whenever he needs to send a message containing the secret key, he uses his pattern encryption oracles. In the second case, the secret key is shared with a dishonest agent, $\mathcal{B}$ generates a new pair of key with $\mathcal{KG}(\eta)$ and will use this fresh keys.

When $\mathcal{B}$ receives a message encrypted with a fresh key, if it is a challenge key, the case is similar other challenge keys, otherwise either $\mathcal{B}$ has generated the key and thus knows its inverse, or the key has been sent by $\mathcal{A}_c$ to an honest agent, and thus is known by $\mathcal{B}$.

The adversary $\mathcal{B}$ can find bit b when he obtains the challenge nonce N_b or a challenge secret key. With this construction, formula 1 is still true.

E Proof of Theorem 3

The proof is very similar to the previous one. Using the same notations, we build two adversaries: $\mathcal{B}_1$, a Q-PAT-UNF-IND-CCA$_1$ adversary and $\mathcal{B}_2$, a Q-PAT-UNF-IND-CCA$_2$ adversary such that:

$$Pr\left[t_c \overset{R}{\leftarrow} \mathcal{A}_c/\eta, \Pi\ ;\ t_c\ NDY\right] \leq Q(\eta)^2.n_N.\mathbf{Adv}^{Q-PUI_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{B}_1}(\eta) + \mathbf{Adv}^{Q-PUI_2}_{(\mathcal{AE},\mathcal{SS}),\mathcal{B}_2}(\eta) \quad (2)$$

Adversaries $\mathcal{B}_1$ and B_2 are similar to $\mathcal{B}$ in the previous proof. $\mathcal{B}_1$ chooses a nonce $H.s.N$, gives it two values N_0 and N_1 and tries to obtain the value of its challenge bit b, $\mathcal{B}_2$ just uses $\mathcal{A}_c$ and waits for a newly created signature. Emission and reception of messages are made as in the previous proof using oracles.

The main difference is that a message can be NDY because it contains a signature. If the first NDY message contains a signature that always appeared encrypted by the public key of an honest agent, then this signature contains a nonce. If this nonce is the nonce chosen by $\mathcal{B}_1$, then $\mathcal{B}_1$ obtains the bit b. Else, if this is a new signature, $\mathcal{B}_2$ wins its challenge. From then, it is easy to deduce formula 2.

Nonces are Probably Different

We consider that anytime a computational adversary picks up some nonces, they are different one from another. The adversary can only get a number m of nonces that is polynomial in η and we suppose that the number n of possible nonces is exponential in η (so $m < n$). Let P be the probability that the adversary gets two times the same nonces.

$$1 - P = \frac{n}{n}\frac{n-1}{n}...\frac{n-(m-1)}{n}$$

Thus, we have the following inequalities:

$$0 \leq P \leq 1 - \big(1 - \frac{m-1}{n}\big)^m$$

Proposition E.1 *For any $x \in [0,1[$ and $a \geq 1$,*

$$\big(1-x\big)^a \geq 1 - x.a$$

Proof: Consider the function $f(x) = \big(1-x\big)^a - 1 + x.a$. Derive it twice to get the result. ■ Applying the proposition, we get:

$$0 \leq P \leq \frac{m.(m-1)}{n}$$

As m is polynomial and n is exponential in η, P is negligible in η. When considering an adversary that has a non-negligible advantage against something, it still has its advantage if we consider only executions where nonces are distinct.

Experiment $\mathbf{Exp}^{N-PUI_2}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta)$:
$b \xleftarrow{R} [0,1]$
for $i = 1$ **to** N **do**
$(pk_i, sk_i) \xleftarrow{R} \mathcal{KG}(\eta)$
$(sik_i, vk_i) \xleftarrow{R} \mathcal{KG}'(\eta)$
$(m, \sigma) \xleftarrow{R} \mathcal{A}/\eta, pk_1, ..., pk_N,$
$vk_1, ..., vk_N,$
$\mathcal{E}^{Pat}_{pk_1}(LR(.,.,b)), ..., \mathcal{E}^{Pat}_{pk_N}(LR(.,.,b)),$
$\mathcal{D}_{sk_1}(.), ..., \mathcal{D}_{sk_N}(.))$
$\mathcal{S}_{sik_1}(.), ..., \mathcal{S}_{sik_N}(.),$
$\mathcal{V}_{vk_1}(.,.), ..., \mathcal{V}_{vk_N}(.,.)$
if σ is a valid signature of m under sik_i
not produced by $\mathcal{S}_{sik_i}(.)$
return 1
else return 0

Experiment $\mathbf{Exp}^{N-PUI_1}_{(\mathcal{AE},\mathcal{SS}),\mathcal{A}}(\eta, b)$:
for $i = 1$ **to** N **do**
$(pk_i, sk_i) \xleftarrow{R} \mathcal{KG}(\eta)$
$(sik_i, vk_i) \xleftarrow{R} \mathcal{KG}'(\eta)$
$b \xleftarrow{R} \mathcal{A}/\eta, pk_1, ..., pk_N,$
$vk_1, ..., vk_N,$
$\mathcal{E}^{Pat}_{pk_1}(LR(.,.,b)), ..., \mathcal{E}^{Pat}_{pk_N}(LR(.,.,b)),$
$\mathcal{D}_{sk_1}(.), ..., \mathcal{D}_{sk_N}(.))$
$\mathcal{S}_{sik_1}(.), ..., \mathcal{S}_{sik_N}(.),$
$\mathcal{V}_{vk_1}(.,.), ..., \mathcal{V}_{vk_N}(.,.)$
return b

Figure 2: N-PAT-UNF-IND-CCA Experiments