(De)Compositions of Cryptographic Schemes and their Applications to Protocols

R. Janvier, Y. Lakhnech and L. Mazaré

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

June 10, 2005

Abstract: The main result of this paper is that the Dolev-Yao model is a safe abstraction of the computational model for security protocols including those that combine asymmetric and symmetric encryption, signature and hashing. Moreover, message forwarding and private key transmission are allowed. To our knowledge this is the first result that deals with hash functions and the combination of these cryptographic primitives.

A key step towards this result is a general definition of correction of cryptographic primitives, that unifies well known correctness criteria such as IND-CPA, IND-CCA, unforgeability etc.... and a theorem that allows to reduce the correctness of a composition of two cryptographic schemes to the correctness of each one.

This updated version contains a new, simpler, proof for the reduction theorems.

Introduction

Historically, verification of cryptographic protocols has been separated in two distinct branches. In the *symbolic approach*, originating from the work of Dolev and Yao [14], cryptographic primitives are viewed as functions on a space of symbolic terms; while in the *computational approach* they are viewed as possibly randomized functions on bit strings.

A rich collection of *automatic* verification methods and tools have been developed [27, 11, 29, 18, 10, 15] in the symbolic approach. They rely upon the perfect cryptography assumption which can be roughly summarized as follows: messages are represented as algebraic terms, nonces are represented as names and fresh nonce creation is perfect, that is, nonces range over an infinite domain of names and each nonce creation yields a different name, the same holds for keys. Moreover, no information can be extracted from an encrypted message unless the inverse of the key used to encrypt the message is known. In this approach there is a single attacker that is modeled as an infinite process without bounds on its computational resources.

In the *computational approach*, cryptographic primitives operate on strings of bits and their security is defined in terms of high complexity and weak probability of success [16, 7] of any attacker. Protocols as well as attackers are randomized polynomial-time Turing machines. This computational approach is recognized as more realistic than the symbolic approach. However, its complexity makes it very difficult to design automatic verification tools.

Therefore, results of the type:

> *If protocol Π uses the cryptographic schemes $S_1, \cdots, S_N$, if each scheme S_i is correct with respect to the security notion C_i then correctness of the protocol established in the symbolic model implies its correctness in the computational one.*

are of extreme importance for gaining confidence that a cryptographic protocol is secure. We call this type of results *soundness results of the symbolic approach*.

In this paper, we present a soundness result for protocols with asymmetric and symmetric encryption, signature and hashing. We emphasize that the main difficulty here is the combination of these primitives.

The main step to get this result is the introduction of a security criterion that allows us to combine asymmetric and symmetric key cryptography as well as signature and hashing. To understand what is going on, imagine a cryptographic library that offers these different kinds of primitives. What does it mean that this library is secure? A priori it is not clear whether it is sufficient to say that each primitive is secure when taken on its own. There might be some unexpected effects when for instance the encryption of a signed message is hashed!

To answer this question we prove a powerful reduction theorem for security criteria. Typically, this theorem allows us to prove results of the form: if the cryptographic scheme S_1 (resp. S_2) satisfies the criterion C_1 (resp. C_2) then their combination satisfies criterion C, where C is some combination of C_1 and C_2. Then, we introduce a security criterion for cryptographic libraries as above and use the reduction theorem to relate our security criterion to existing ones, namely IND-CCA, selective forgery against adaptive chosen-message attack and collision resistance.

Related work In the last years, effort has been invested to bridge the gap between the symbolic and computational approaches. In their ground-breaking paper [2] Abadi and Rogaway prove that message *indistinguishability* in the symbolic model is valid in the computational model when making some assumptions on the encryption scheme. In this and subsequent papers [1, 20, 26], it is showed that if two messages are not distinguishable in the symbolic model, then their computational interpretations cannot be separated by a Turing machine in a reasonable (polynomial) time. These papers deal with passive attackers that do not intervene during protocol execution. Active attackers are considered in [30, 25, 4, 24, 13, 19, 12, 21]. Backes, Pflitzman and Waidner developed a Dolev-Yao-style cryptographic library with a provably correct implementation [4, 5, 3]. The security property considered there, called *reactive simulatability*, is a very attractive and powerful notion which is robust respect to general composition. Canetti and Herzog demonstrate in [12] how Dolev-Yao style symbolic analysis can be used to assert the security of cryptographic protocols within the universally composable security framework. This framework also allows for strong composability properties. Soundness of the symbolic approach for public key encryption is considered in [30, 25]. Asymmetric encryption and digital signature are considered in [13, 21].

Compared to our paper [21], we improve with respect to the following: 1) in [21] we only consider asymmetric encryption and digital signature and 2) we substantially generalize the reduction theorem to be applicable to asymmetric and symmetric encryption, digital signature and hashing. Compared to previous versions of this paper [22], we clarify the proof of the reduction theorem.

Paper organization The next section gives the necessary preliminaries related to the computational model. In the following section, we generalize and simplify the notion of security criterion and apply it to asymmetric encryption, signature, symmetric encryption, hashing and a mix of all these primitives. Section 3 formulates the reduction theorem. Then, this theorem is applied to relate the combined security criterion to the simple ones. Section 4 uses these results to show that, under some quite nonrestrictive hypotheses, the symbolic model is a safe abstraction of the computational model. Finally, some concluding remarks are drawn.

1 Preliminaries

1.1 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 secret key and a cipher text 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 to be polynomially bounded by η.

A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined as above except that $\mathcal{KG}$ generates one key instead of a pair, and hence, we require $\mathcal{D}(\mathcal{E}(m, k), k) = m$.

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 $\mathcal{KG}$. We say that σ is a valid signature under sik if there exists m such that $\mathcal{V}(m, \sigma, vk)$ returns true. We still assume that the algorithms have a polynomial complexity.

A *hashing algorithm* is a polynomial deterministic algorithm that, given a key k and a bit-string bs, computes another bit-string of size η. The key generation algorithm is not important and one can suppose that k is chosen randomly among strings of size η.

1.2 Randomized Turing Machines with Oracle

An adversary for a given scheme is a Polynomial Random Turing Machine (PRTM) which has access to an oracle. In the following, we consider Turing machines whose execution is polynomially bounded in the security parameter η, i.e. there exists a polynomial P such that for any input corresponding to security parameter η, the machine stops within $P(\eta)$ steps.

To model access to the oracle, 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 the results. Then, let $\underline{F}$ be a new action. We define our PRTM as a pair of a Turing machine $\mathcal{A}$ that can use transition $\underline{F}$ and another Turing machine F representing the oracle. F can also be described by a PRTM (which can also access oracles). The semantics of $\mathcal{A}/F$ is the standard semantics of $\mathcal{A}$ except that whenever $\mathcal{A}$ fires the action $\underline{F}$, F is executed with the arguments tape as input and the results tape as output.

It is possible to encode access to multiple oracles using F (by giving in the arguments tape the name of the chosen oracle). Hence, to simplify notations, we directly write $\mathcal{A}/f_1, ..., f_n$ where f_i are PRTM and oracles are called using the $\underline{f_i}$ action when defining $\mathcal{A}$.

A function $g : \mathbb{R} \to \mathbb{R}$ 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 $|g(x)| < x^{-c}$, for all $x > N_c$.

2 Security Criteria

A security criterion is defined as an experiment involving an adversary (represented by a PRTM). The experiment proceeds as follows. First some parameters θ are generated randomly. The adversary is executed and can use an oracle F which depends on θ. At the end, the adversary has to answer a string of bits which is verified by an algorithm V which also uses θ (e.g. θ includes a bit b and the adversary has to output the value of b).

2.1 Security Criterion

A criterion γ is a triple $(\Theta; F; V)$ where

- Θ is a PRTM that randomly generates some challenge θ (for example, a bit b and a pair of keys (pk, sk)).
- F is a PRTM that takes as arguments a string of bits s and a challenge θ and outputs a new string of bits. F represents the oracles that an adversary can call to solve its challenge.
- V is a PRTM that takes as arguments a string of bits s and a challenge θ and outputs either true or false. It represents the verification made on the result computed by the adversary. The answer true (resp. false) means that the adversary solved (resp. did not solve) the challenge.

Note that Θ can generate an arbitrary number of parameters and F can represent an arbitrary number of oracles. Thus, it is possible to define criteria with multiples Θ and F. As soon as there is no risk for ambiguity, we use the same notation for the challenge generator Θ and the generated challenge θ (both are denoted using θ).

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

$$\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) = 2.\big(Pr[\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta) = true] - PrRand^{\gamma}\big)$$

where $\mathbf{Exp}$ is the Turing machine defined by:

```
Experiment $\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta)$:
    $\theta \leftarrow \Theta(\eta)$
    $d \leftarrow \mathcal{A}/\eta, \lambda s.F(s, \theta)$
    return $V(d, \theta)$
```

and $PrRand^{\gamma}$ is the best probability to solve the challenge that an adversary can have without using oracle F. Formally, $PrRand^{\gamma}$ is the maximum of $Pr[\mathbf{Exp'}^{\gamma}_{\mathcal{A}}(\eta) = true]$ where $\mathcal{A}$ ranges over any possible PRTM and $\mathbf{Exp'}$ is similar to $\mathbf{Exp}$ except that F cannot be used by $\mathcal{A}$.

```
Experiment $\mathbf{Exp'}^{\gamma}_{\mathcal{A}}(\eta)$:
    $\theta \leftarrow \Theta(\eta)$
    $d \leftarrow \mathcal{A}/\eta$
    return $V(d, \theta)$
```

2.2 The N-PAT-IND-CCA Criterion

We introduce a security criterion that turns out to be useful for protocols where secret keys are exchanged. This definition was first given in [21] where more discussion is available. In the classical N-IND-CCA criterion (see [6] about N-IND-CCA and its reduction to IND-CCA), a random bit b is sampled. For each key, the adversary has access to a left-right oracle (the adversary submits a pair of bit-strings bs_0, bs_1 and receives the encoding of bs_b) and a decryption oracle (that does not work on the outputs of the left-right oracle). The adversary has to guess the value of b.

Since it has no information concerning secret keys, the adversary cannot get the encryption of a challenge secret key under a challenge public key. Therefore, we introduce N-PAT-IND-CCA, which allows the adversary to obtain the encryption of messages containing challenge secret keys, even if he 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.

Pattern terms are terms where new atomic constants have been added: pattern variables. These variables represent the different challenge secret keys and are denoted by $[i]$ (this asks the oracle to replace the pattern variable by the value of sk_i). Variables can be used as atomic messages (data pattern) or at a key position (key pattern). When a left-right oracle is given a pattern term, it replaces patterns by values of corresponding keys and encodes the so-obtained message. More formally, patterns are given by the following grammar where bs is a bit-string and i is an integer. In the definition of pattern terms, we use the following binary operators : concatenation, encryption and signature. Concatenation of patterns pat_0 and pat_1 is written $\langle pat_0, pat_1\rangle$. Encryption of pat with key bs is denoted by $\{pat\}_{bs}$. Signature of pat with key bs is denoted by $sig(pat, bs)$. Similarly, when the key is a challenge key, it is represented by a pattern variable $[i]$. Finally, one unary operator, hashing, is defined over patterns and is denoted by h.

$$pat \quad ::= \quad \langle pat, pat\rangle \mid \{pat\}_{bs} \mid \{pat\}_{[i]} \mid bs \mid [i] \mid sig(pat, [i]) \mid sig(pat, bs) \mid h(pat)$$

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

$$\begin{aligned} v(bs,\theta) &= bs \\ v([i],\theta) &= sk_i \\ v(\langle p_1,p_2\rangle,\theta) &= v(p_1,\theta).v(p_2,\theta) \\ v(\{p\}_{[i]},\theta) &= \mathcal{E}(v(p,\theta),pk_i) \end{aligned} \qquad \begin{aligned} v(\{p\}_{bs},\theta) &= \mathcal{E}(v(p,\theta),bs) \\ v(sig(p,bs),\theta) &= \mathcal{S}(v(p,\theta),bs) \\ v(sig(p,[i]),\theta) &= \mathcal{S}(v(p,\theta),sk_i) \\ v(h(p),\theta) &= \mathcal{H}(k,v(p,\theta)) \end{aligned}$$

There is yet a restriction. Keys are ordered and a pattern $[i]$ can only be encrypted under pk_j if $i > j$. This restriction is well-known in cryptography and widely accepted. When the left-right pattern encryption oracle related to key i is given two pattern 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 : $v(pat_b, \theta)$ encoded by pk_i. To win, the adversary has to guess the value of secret bit b. Note that an adversary can submit arguments of different length to the left-right oracle but this does not create any problem (an interesting discussion on that point appears in [2]).

Henceforth, let $\mathcal{AE}$ be an asymmetric encryption scheme. Then, criterion N-PAT-IND-CCA is given by $\gamma_N = (\Theta; F; V)$, where Θ randomly generates N pairs of keys using $\mathcal{KG}$ and a bit b; V verifies that the adversary gave the right value for bit b; and F gives access to three oracles for each i : a left-right encryption oracle that takes as argument a pair of patterns $\langle pat_0, pat_1\rangle$ and outputs pat_b completed with the secret keys ($v(pat_b, \theta)$) and encoded using pk_i; a decryption oracle that decodes any message not produced by the former encryption oracle; and an oracle that simply makes the public key available.

Then, $\mathcal{AE}$ is said N-PAT-IND-CCA iff for any adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}_{\mathcal{A}}^{\gamma_N}(\eta)$ is negligible. Note that N-PAT-IND-CCA with $N = 1$ corresponds to IND-CCA.

2.3 The N-UNF Criterion

The N-UNF criterion is an extension of Selective Forgery Against Adaptive Chosen-Message Attacks to the case of N different keys (a good survey on properties for signature schemes is available in [17]). It was also already defined in [21]. Here, we rephrase this definition to put it in the shape of our new criterion formalization.

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 adversary $\mathcal{A}$ is given N verification keys and has to produce a message and its signature under one of the keys. It is also given the security parameter η and N signature oracles $\mathcal{S}_{sik_i}(.)$.

Let $\mathcal{SS}$ be a signature scheme. The N-UNF criterion is given by $\gamma_N = (\Theta, F, V)$, where Θ generates N signature key pairs using the key generation algorithm from $\mathcal{SS}$. F permits the access to two oracles for each signature key pair: the first one allows to sign any string of bits; the second one gives the verification key. Verifier V checks that the output of the adversary is a pair containing a message and its signature. This signature must not have been produced by the signature oracle.

An adversary wins against N-UNF when it succeeds in producing a message and its signature. Formally, $\mathcal{SS}$ is said N-UNF, if for any adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{N-UNF}(\eta)$ is negligible. When $N = 1$, N-UNF can be written UNF.

2.4 The N-PAT-SYM-CCA Criterion

A symmetric encryption scheme includes both aspects indistinguishability and authentication that are present in asymmetric encryption and message signature respectively. We reformulate it using our criteria framework in order to apply our reduction theorem. That is, our criterion for symmetric encryption is a combination of IND-CCA and UNF. Indeed, a symmetric encryption should be secure in two ways. The first one is related to IND-CCA, any PRTM should not be able to guess any information from messages encoded with an unknown key. The second one is related to UNF; any PRTM should not be able to forge an encoding without knowing the key (the encrypted message is authenticated). Hence, oracles are similar to those presented in IND-CCA (except that no oracles output the public key), but there are two different ways to win the challenge. The hypothesis of acyclicity regarding keys still holds: k_i can only appear encoded by k_j if $i > j$. The N-PAT-SYM-CCA criterion is $\gamma_N = (\Theta, F, V)$ where Θ generates N symmetric keys and a bit b; F gives access to two oracles for each key: a left-right encryption oracle that takes as argument a pair of patterns $\langle pat_0, pat_1\rangle$ and outputs pat_b completed with the secret keys $(v(pat_b, \theta))$ and encoded with k_i; a decryption oracle that decodes any message not produced by the former encryption oracle. Finally, V is composed of two parts: V_{IND} returns true when the adversary returns bit b; V_{UNF} returns true when the adversary outputs a message encoded by one of the symmetric key and this message has not been produced by an encryption oracle. Then V is satisfied if V_{IND} or V_{UNF} is satisfied. We require that there is no string that satisfies both V_{IND} and V_{UNF} (this can be done by asking the name of the challenge together with its solution to the adversary). The criterion related to IND (Θ, F, V_{IND}) (resp. to UNF (Θ, F, V_{UNF})) is denoted by N-PAT-SYM-CCA/IND (resp. N-PAT-SYM-CCA/UNF).

A symmetric encryption scheme $\mathcal{SE}$ is said N-PAT-SYM-CCA iff for any adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\gamma_N}(\eta)$ is negligible, where γ_N is a criterion including the oracles explained above.

Existence of a 1-PAT-SYM-CCA encryption scheme can be proved under the assumption that there exists an IND-CCA asymmetric encryption scheme and an UNF signature scheme (see appendix A). The 1-PAT-SYM-CCA criterion is equivalent to the authenticated encryption criterion IND-CPA $\wedge$ INT-CTXT which is the strongest notion introduced in [8] for authenticated encryption.

2.5 The HASH Criterion

The HASH criterion is a combination of an IND-CCA criterion, an UNF criterion and a collision free criterion. A hashing algorithm needs to verify three properties to be secure. First an adversary cannot obtain information on a bit-string bs when looking at $\mathcal{H}(k, bs)$. The second property is that if an adversary does not know a bit-string bs, it cannot produce $\mathcal{H}(k, bs)$ even if it knows hashing of messages similar to bs. Finally, it must be hard for an adversary to find two different messages which have the same hash for a given key. More details about criteria related to HASH can be found in [9].

The HASH criterion is $\gamma = (\Theta, F, V)$, where Θ generates a bit b, a key k and a random bit-string N^H of size η. Oracle F gives access to two oracles: an oracle which gives the value of key k and a left-right hashing oracle which takes as input a pair $\langle pat_0, pat_1\rangle$ of hollow patterns (these patterns can ask for inclusion of N^H and have to ask for it at one position at least) and outputs $\mathcal{H}(k, pat_b[N^H])$. Moreover, each pattern can only be submitted once to this oracle in order to avoid guessing attacks. Verifier V is the disjunction of three parts: V_{IND} returns true if the adversary outputs the challenge bit b; V_{UNF} returns true if the adversary outputs a pair $\langle h, pat\rangle$ such that $h = \mathcal{H}(k, pat[N^H])$ and h was not produced by F; V_{CF} returns true if the adversary outputs a pair $\langle bs_0, bs_1\rangle$ such that $\mathcal{H}(k, bs_0) = \mathcal{H}(k, bs_1)$, and bit-strings bs_0 and bs_1 are different.

A hashing algorithm is said HASH iff for any adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}_{\mathcal{A}}^{\gamma_H}(\eta)$ is negligible.

The criterion related to IND (Θ, F, V_{IND}) (resp. to UNF (Θ, F, V_{UNF})) is denoted by HASH/IND (resp. HASH/UNF). The last criterion related to collision free is denoted HASH/CF.

Proposition 2.1 *If an algorithm $\mathcal{H}$ is secure against HASH/IND and HASH/CF and $PrRand^{CF}$ and $PrRand^{UNF}$ are negligible, then $\mathcal{H}$ verifies HASH/UNF and so is secure against HASH.*

Proof: This proof is detailed in appendix B. ■

Let us spend a few words explaining our requirements on hashing algorithms: indistinguishability and collision freeness. In the cryptographic literature, one usually finds one-wayness [1] and collision freeness as requirements. We require, however, indistinguishability instead of one-wayness. This is because, exactly as for asymmetric encryption, one-wayness is too weak as it should not be possible to infer any information on m just by looking at $h(m)$.

Note that it is not clear to us whether there exists an algorithm satisfying our requirements. However our requirements seem necessary to prove soundness of the symbolic model.

2.6 Mixing all Criteria

Let us now consider an encryption scheme, or rather a cryptographic library, that includes the cryptographic primitives above, i.e., asymmetric encryption, symmetric encryption, signature and hashing. The security of such a library can be defined as a game, where an adversary has access to each of the oracles above and wins the game, if it succeeds to guess the value of the bit b, forge a signature, forge an encryption by a symmetric key, or construct a hash-collision. There are some restriction on the patterns the adversary can use. The restriction essentially forbids cycles, as in the case of asymmetric encryption. We say that a cryptographic library satisfies the N-PASSH-CCA criterion, if the advantage of any adversary against this combined criterion is negligible. More formally, we have the following N-PASSH-CCA criterion: $\gamma = (\Theta, F, V)$ where Θ is composed of four parts:

- Θ_a generates N pairs of asymmetric keys (pk_1, sk_1) to (pk_N, sk_N).
- Θ_b generates N symmetric keys k_1 to k_N.
- Θ_c generates N pairs of signature keys (sik_1, vk_1) to (sik_N, vk_N).
- Θ_d generates a nonce N^H, a key k as well as a challenge bit b.

F is also split in four parts:

- F_a corresponds to the oracles using θ_a as in N-PAT-IND-CCA except that patterns can also ask for symmetric encryption, symmetric keys, signature of a message, signature keys, hashing of a message and nonce N^H. F_a depends on θ_a, θ_b, θ_c and θ_d.
- F_b corresponds to oracles using θ_b as in N-PAT-SYM-CCA, patterns are also extended but cannot include asymmetric keys from θ_a. F_b depends on θ_b, θ_c and θ_d.
- F_c corresponds to oracles using θ_c as in N-UNF, F_c depends only on θ_c.
- F_d corresponds to oracles using θ_d as in HASH, F_d depends only on θ_c.

Finally V is also a disjunction of five parts:

- V_{IND} answers true if its argument if the bit b in Θ_d.
- $V_{UNF-SYM}$ answers true if it receives a symmetric encryption not forged by F_b.
- $V_{UNF-SIGN}$ answers true if it receives a signature not forged by F_c.
- $V_{UNF-HASH}$ answers true if it receives a pair h, pat where $h = \mathcal{H}(k, v(pat, N^H)$ and h has not been forged using F_d.
- $V_{CF-HASH}$ answers true if it receives a pair of distinct bit-strings bs_0, bs_1 that have the same hash.

Let us define the N-PAT-ASYM-SYM-SIGN-HASH-CCA (N-PASSH-CCA) criterion as $\gamma = (\Theta, F, V)$ where Θ is composed of four parts:

[1] Intuitively, a function f is one-way, if given $f(x)$ but not x it is hard to find a value y such that $f(y) = f(x)$.

- Θ_a generates N pairs of asymmetric keys (pk_1, sk_1) to (pk_N, sk_N).
- Θ_b generates N symmetric keys k_1 to k_N.
- Θ_c generates N pairs of signature keys (sik_1, vk_1) to (sik_N, vk_N).
- Θ_d generates a nonce N^H, a key k as well as a challenge bit b.

F is also split in four parts:

- F_a corresponds to the oracles using θ_a as in N-PAT-IND-CCA except that patterns can also ask for symmetric encryption, symmetric keys, signature of a message, signature keys, hashing of a message and nonce N^H. F_a depends on θ_a, θ_b, θ_c and θ_d.
- F_b corresponds to oracles using θ_b as in N-PAT-SYM-CCA, patterns are also extended but cannot include asymmetric keys from θ_a. F_b depends on θ_b, θ_c and θ_d.
- F_c corresponds to oracles using θ_c as in N-UNF, F_c depends only on θ_c.
- F_d corresponds to oracles using θ_d as in HASH, F_d depends only on θ_c.

Finally V is also a disjunction of five parts:

- V_{IND} answers true if its argument if the bit b in Θ_d.
- $V_{UNF-SYM}$ answers true if it receives a symmetric encryption not forged by F_b.
- $V_{UNF-SIGN}$ answers true if it receives a signature not forged by F_c.
- $V_{UNF-HASH}$ answers true if it receives a pair h, pat where $h = \mathcal{H}(k, v(pat, N^H)$ and h has not been forged using F_d.
- $V_{CF-HASH}$ answers true if it receives a pair of distinct bit-strings bs_0, bs_1 that have the same hash.

3 Reductions of Criteria

In this section, we present a generic result allowing 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.

This result can be seen as a tool for proving that a criterion γ is at least as secure as a criterion γ' but also allows to decompose and split a criterion into simpler ones. We begin by presenting a simple version of the theorem.

3.1 Criterion Partition and the Reduction Theorems

Let $\gamma = (\theta_1, \theta_2; F_1, F_2; V_2)$ be a criterion. Let γ_1 and γ_2 be two criteria such that:

- There exist two PRTM G and H such that:

$$\begin{aligned} G(H(s, \theta_2, \theta_2'), 1, \theta_1) &= F_1(s, \theta_1, \theta_2) \quad (1) \\ G(H(s, \theta_2, \theta_2'), 0, \theta_1) &= F_1(s, \theta_1, \theta_2') \quad (2) \end{aligned}$$

  Oracle G operates on a string of bits, thus it must receive two challenge information, a bit b and θ_1.

- $\gamma_2 = (\theta_2; F_2; V_2)$ and $\gamma_1 = (b, \theta_1; G; verif_b)$ where b generates a random bit and $verif_b$ is the PRTM verifying that the output of the adversary is b: $verif_b(s, b, \theta_1) = (s \Leftrightarrow b)$.
- $F_2(s, \theta_1, \theta_2)$ and $V_2(s, \theta_1, \theta_2)$ do not depend on θ_1.

Then we say that (γ_1, γ_2) is a *valid simplified partition* of γ.

Theorem 3.1 (Simplified Reduction Theorem) *Let (γ_1, γ_2) be a valid simplified partition of γ. For any PRTM $\mathcal{A}$, there exist two PRTM $\mathcal{A}^o$ and $\mathcal{B}$ such that*

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)|$$

The proof appears in appendix C. Notice that in applying the reduction theorem above, the difficulty is not to find arbitrary functions G and H that satisfy the Equations (1) and (2) but rather to find such functions that induce criteria γ_1 and γ_2 with negligible corresponding advantages.

The applicability of the simplified reduction theorem is restricted by the fact that the verfication algorithm V only depends θ_2. We show now that we can avoid this restriction. So let us assume that the PRTM V is represented two PRTM's V_1 and V_2 such that V_1 (resp. V_2) depends only on θ_1 (resp. θ_2) and V returns true if V_1 or V_2 returns true. By abuse of notation we write $V_1 \vee V_2$ to underpin this. The criteria γ_1 and γ_2 are defined as above but now a new criterion $\gamma_3 = (b, \theta_1; G; V_1)$ occurs in the partition. Then, we say that $(\gamma_1, \gamma_2, \gamma_3)$ is a valid partition of γ, if there is no string s such that V_1 and V_2 are both verified on s (intuitively, the adversary should know which part of the challenge he is trying to win).

Theorem 3.2 (Reduction Theorem) *Let $(\gamma_1, \gamma_2, \gamma_3)$ be a valid partition of γ. For any PRTM $\mathcal{A}$, there exist three PRTM $\mathcal{A}^o$, $\mathcal{A}^1$ and $\mathcal{B}$ such that*

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)| + |\mathbf{Adv}^{\gamma_3}_{\mathcal{A}^1}(\eta)|$$

3.2 Applications of the Reduction Theorems

This section contains application examples of our reduction theorems. These applications are mainly useful for composition of security criteria.

The first proposition (which was already given in [21]) states that N-PAT-IND-CCA is equivalent to IND-CCA. This proposition is useful as the criterion is well-studied in the literature and as there are algorithms proven to be IND-CCA.

Proposition 3.1 *([21]) If an encryption scheme is secure against IND-CCA, then it is secure against N-PAT-IND-CCA for any N.*

Proofs for all the proposition in this section appear in appendix D

Proposition 3.2 *If a symmetric encryption scheme is secure against SYM-CCA/IND and SYM-CCA/UNF, then it is secure against N-PAT-SYM-CCA for any N.*

The following proposition states that the combination of secure encryption schemes is a secure encryption scheme. In other words, combining secure encryption schemes is harmless as long as cycles are avoided.

Proposition 3.3 *If an asymmetric encryption scheme $\mathcal{AE}$ is IND-CCA, a symmetric encryption scheme $\mathcal{SE}$ is SYM-CCA, a signature scheme $\mathcal{SS}$ is UNF and a hashing algorithm $\mathcal{H}$ is HASH, then the composition $(\mathcal{AE}, \mathcal{SE}, \mathcal{SS}, \mathcal{H})$ is N-PASSH-CCA.*

Proof: We only present here the first step of the proof, the other steps are similar. Let Θ_1 be (Θ_a, Θ_b) and Θ_2 be (Θ_c, Θ_d). In the same way, F_1 (resp. F_2) can be used to access F_a and F_b (resp. F_c and F_d). V_1 is $V_{UNF-SYM}$ and V_2 is the disjunction of $V_{UNF-SIGN}$, $V_{UNF-HASH}$, $V_{CF-HASH}$ and V_{IND}. H is defined as above for IND-CCA or UNF and G is also defined as above for encryption, decryption (asymmetric and symmetric keys) and public key. F_1, F_2, V_1 and V_2 depend on the right parameters hence we define a valid partition of γ. The reduction theorem gives that for any PRTM $\mathcal{A}$, there exist three PRTM $\mathcal{B}$, $\mathcal{A}^o$ and $\mathcal{A}^1$ such that:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)| + |\mathbf{Adv}^{\gamma_3}_{\mathcal{A}^1}(\eta)|$$

Criteria γ_1, γ_2 and γ_3 can easily be partitioned in a similar way to get the conclusion. ■

3.3 Unbounded Number of Challenges

We want to consider the case where the number of challenges is not bounded any more like in N-IND-CCA where only N keys are generated for any η. For that purpose, criteria are extended to a polynomial number of challenges. For example, if P is a polynomial, then the P-IND-CCA criterion uses $P(\eta)$ keys. The objective here is to generalize the previous results to this case.

Proposition 3.4 *Let P and Q be two polynomials from $\mathbb{N}[X]$. Let D be a PRTM that given an integer i returns C_i, a PRTM whose execution takes less than $Q(\eta)$ steps. If the execution of D also takes less than $Q(\eta)$ steps, then for any criterion γ, there exists a PRTM C whose execution takes less than $2.Q(\eta) + P(\eta)$ steps such that for any η:*

$$\mathbf{Adv}^{\gamma}_{C}(\eta) = \frac{1}{P(\eta)} \sum_{i=1}^{P(\eta)} \mathbf{Adv}^{\gamma}_{C_i}(\eta)$$

Adversary C randomly chooses the PRTM C_i that it is going to use and executes it.

```
Adversary $C$:
    $r \leftarrow [1..P(\eta)]$
    $C_r \leftarrow D/r$
    $d \leftarrow C_r/\eta$
    return $d$
```

This property allows us to consider the case of a polynomial number of challenge (and also the case of an unbounded number of challenges as only a finite part of them can be used). If the advantage of any PRTM $\mathcal{A}$ against γ_P is the sum of the advantages of $P(\eta)$ PRTM against γ. Then if each of the latest PRTM are bounded in time using a same polynomial Q, the advantage of $\mathcal{A}$ is also equal (modulo a division by $P(\eta)$) to the advantage of a PRTM against γ. Hence, if the considered scheme is secure against γ, it is also secure against γ_P.

This method applies on all the previous applications of our reduction theorems. Hence, we have:

Proposition 3.5 *If an encryption scheme is secure against IND-CCA, then it is secure against P-IND-CCA for any polynomial P.*

If a symmetric encryption scheme is secure against SYM-CCA/IND and SYM-CCA/UNF, then it is secure against P-PAT-SYM-CCA for any polynomial P.

If an asymmetric encryption scheme $\mathcal{AE}$ is IND-CCA, a symmetric encryption scheme $\mathcal{SE}$ is SYM-CCA, a signature scheme $\mathcal{SS}$ is UNF and a hashing algorithm $\mathcal{H}$ is HASH, then the composition $(\mathcal{AE}, \mathcal{SE}, \mathcal{SS}, \mathcal{H})$ is P-PASSH-CCA for any polynomial P.

4 Dolev-Yao is a Safe Abstraction

4.1 Definitions for the Symbolic Model

In this section, we give the basic definitions that are used to introduce the symbolic aspects of protocol checking. Symbolic 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 $Nonces$ are usually 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, symmetric encryption keys and signature keys. There is a unary function over $Keys$ associating each key k to its inverse k^{-1} such that $k = (k^{-1})^{-1}$. For symmetric encryption, the inverse of a key is itself: $k = k^{-1}$. The following binary operators are defined over messages: concatenation, encryption and signature. Concatenation of messages m and n is written $\langle m, n\rangle$. Encryption of message m with key k is denoted by $\{m\}_k$. Signature of message m with key k is denoted by $sig(m,k)$. Finally, one unary operator, hashing, is defined over messages and is denoted by h.

Next, we recall the definition of the *entailment* relation $E \vdash m$ (introduced in [14]) where E is a finite set of messages and m a message. Intuitively, $E \vdash m$ means that m can be deduced from the set of messages E. 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 $E \vdash m$ and $E \vdash k$ then $E \vdash sig(m, k)$.
- 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 $E \vdash m$.
- If $E \vdash sig(m, k)$ then $E \vdash m$.
- If $E \vdash m$, then $E \vdash h(m)$.

4.2 Symbolic and Computational Semantics

For the sake of presentation, we consider protocols that only involve a single role. Moreover, this role is only instantiated in one session. This is done without loss of generality when a bounded number of sessions is considered. Indeed, each interleaving of the actions of the different participants can be seen as a different protocol.

Thus, a protocol is described by a list of actions which are either emission $!m$ or reception $?m$ of a message m. We consider the classical adversary model where the adversary controls the network, receives all the outputs ($!m$) and submits some forged message to the inputs ($?m$).

Henceforth, let us consider an arbitrary fixed protocol $\ddagger_1 t_1 ... \ddagger_k t_k$, where $\ddagger_i$ is either "!" or "?" and t_i is a term. There are two different execution models, one for the symbolic setting and one for the computational setting producing a symbolic and a computational trace, respectively. A *symbolic action sequence* is a list of actions $s\ m$ where s is either $?$ or $!$ and m is a ground (closed) message. A *symbolic trace* is a symbolic action sequence $\ddagger_1 m_1 ... \ddagger_{k'} m_{k'}$ with $k' \leq k$ that satisfies the following conditions:

1. There exists a ground substitution σ such that for any i, $t_i\sigma = m_i$;
2. For any i, if $\ddagger_i$ is "?", then m_i is deducible from the previous messages m_1 to m_{i-1} and the initial knowledge of the adversary E_0,i.e.,
$$E_0, m_1, ..., m_{i-1} \vdash m_i.$$
The set E_0 contains the atomic messages of the m_i's that do not appear in any t_i, i.e. $E_0 = \bigcup_i atoms(m_i) \setminus \bigcup_i atoms(t_i)$.

The set $trace(\Pi)$ contains the possible traces for protocol Π.

A *computational action sequence* is a list of actions $\ddagger\, bs$ where bs is a bit-string and $\ddagger$ is either "?" or "!". A *computational trace* is the result of the interaction of an adversary $\mathcal{A}$, which is a polynomial random Turing machine, and the protocol. This interaction is defined using the Turing machine $Exec(\mathcal{A}, \Pi)$. Since we are interested in relating the symbolic and computational semantics we define $Exec$ in such way that along the computational trace it outputs a corresponding symbolic action sequence. We then show that the symbolic action sequence is a trace except for negligible probability. The reader should be convinced that producing the symbolic action sequence by no means interferes with the computational semantics.

To simplify the presentation of the $Exec$ algorithm, we only give pseudo-code using the following functions:

- $init(\Pi)$ generates the keys and nonces and that are chosen by the protocol Π, i.e., those in $atoms(\Pi)$, and not by the adversary. It returns a substitution θ associating bit-string values to these elements.

- $parse(bs, t, \theta, \sigma)$ parses the bit-string bs using prototype t and knowledge from θ, it returns the updated version of θ as well as an updated symbolic substitution σ.
- $concr(m, \theta)$ concretizes message m using knowledge from θ and returns the corresponding bit-string.
- $compl(\sigma)$ completes the symbolic substitution σ by associating remaining free variables to a distinct fresh nonces.

The $Exec$ algorithm uses two substitutions: the symbolic substitution σ that links protocol variables to messages and the computational substitution that links variables to strings of bits. The adversary can decide to stop interacting with the protocol by providing an answer other than an updated memory mem and a bit string bs when an action $?t$ is to be executed.

```
**Algorithm** $Exec(\mathcal{A}, ‡_1 n_1 ... ‡_k n_k)$:
  $\theta \leftarrow init(‡_1 n_1 ... ‡_k n_k)$
  $mem \leftarrow []$
  **for** $i$ **in** $[1, k]$ **do**
    **if** $‡_i =!$ **then**
      $bs \leftarrow concr(n_i, \theta)$
      $mem \leftarrow \mathcal{A}(bs, mem)$
      $t_c \leftarrow append(‡_i bs, t_c)$
    **else**
      $X \leftarrow \mathcal{A}(mem)$
      **if** $X = bs, mem$ **then**
        $\sigma, \theta \leftarrow parse(bs, m_i, \theta, \sigma)$
        $t_c \leftarrow append(‡_i bs, t_c)$
      **else goto done**
  **done**
  $\sigma \leftarrow compl(\sigma)$
  **return** $(‡_1 m_1 ... ‡_i m_{i-1})\sigma, t_c$
```

The next proposition relates precisely the computational trace and symbolic action sequence that $Exec$ outputs. A computational trace t_c is a *possible concretization* of a symbolic action sequence t_f if there exists a computational substitution θ such that one of the possible valuation of t_f using θ is t_c.

Proposition 4.1 *Let $\mathcal{A}$ be an adversary and Π a protocol. If $Exec(\mathcal{A}, \Pi)$ outputs t_f, t_c, then t_c is a possible concretization of t_f.*

4.3 Relating the Symbolic and Computational Models

The main result of this section is that under some conditions the computational adversary acts as a symbolic adversary with overwhelming probability. This means that the computational adversary, even with all the computing power of polynomial-time random Turing machines, cannot have a behavior not represented by a symbolic adversary.

Hypotheses over Cryptographic Schemes and Protocols

In order to be able to use the former results, the cryptographic schemes used in the implementation of the protocol should verify the following properties.

- The asymmetric encryption scheme $\mathcal{AE}$ used in the protocol is IND-CCA.
- The symmetric encryption scheme $\mathcal{SE}$ is 1-PAT-SYM-CCA and the probability to forge a cyphertext without access to the oracles is negligible.
- The signature scheme $\mathcal{SS}$ is UNF and the probability to forge a signature without access to the oracles is negligible.

- The hashing algorithm $\mathcal{H}$ is HASH and the probability to construct a hash-collision is negligible.

These are also some restrictions on Π that are defined in the symbolic world (as they are easier to check there with automated tools).

- In an asymmetric encoding using pk, anything can appear except secret keys generated before pk (and the secret key related to pk too).
- In a symmetric encoding using k, forbidden messages are any secret keys nor any symmetric keys generated before k.
- In a signature using sik and in any hashed message, there cannot be any secret keys, symmetric keys nor any signature keys.
- The protocol has to be secure for its secret, symmetric and signature keys: using the Dolev-Yao model, these keys related to an honest agent cannot be revealed to an intruder (this assumption is reasonable as a protocol should not reveal any key).
- Each hash message in a session between honest agents contains a nonce that remains secret.

We now formulate the main theorem. It states that if the conditions given above are met, then the probability that a computational trace is NDY is negligible. A less general version of this theorem was first given in [25] but only for public key cryptography and protocols with just one layer of encryption. It was then extended to protocols involving emission of secret keys and signature in [21]. Here we give a more general version of this theorem that combines the main cryptographic primitives: public key and symmetric cryptography, digital signature and hashing. The proof is a reduction argument: Given an adversary $\mathcal{A}$ interacting with the protocol, we construct an adversary $\mathcal{B}$ aginst N-PASSH-CCA such that the probability that $\mathcal{A}$ produces a NDY trace is polynomially bounded by the advantage of $\mathcal{B}$.

Theorem 4.1 *For any concrete adversary $\mathcal{A}$:*

$$pr\big(t_f, t_c \leftarrow Exec(\mathcal{A}, \Pi) \textit{ and } t_f \notin traces(\Pi)\big) \textit{ is negligible}$$

Using this theorem, it is possible to relate symbolic and computational properties. This is easy to achieve for trace properties as shown in [25] and [21], but can also be done for strong secrecy [13]. In this last case, the adversary built in the proof of theorem 4.1 has to be modified.

Conclusion

The main contribution of this paper is a proof of correctness of the Dolev-Yao model for protocols that may combine asymmetric and symmetric encryption schemes, signature schemes as well as hash functions. This is important as there is a number of automatic verification tools for protocols that are based on the Dolev-Yao model. The proof of our theorem induces some restrictions on the protocols that are in practice easily met.

As future work, it would be of interest to investigate whether correctness of Dolev-Yao can be proved under weaker assumptions on the cryptographic primitives. Moreover, it would be significant to extend this result to other security properties. A proof of the soundness of the symbolic model when Diffie-Hellman exponentiation is considered is given in [23].

References

[1] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Theoretical Aspects of Computer Software, 4th International Symposium*, volume 2215 of *Lecture Notes in Computer Science*, pages 82–94. Springer, 2001. (document)

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22, Sendai, Japan, 2000. (document), 2.2

[3] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable dolev-yao style cryptographic library. In *CSFW*, pages 204–218. IEEE Computer Society, 2004. (document)

[4] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 220–230, 2003. (document)

[5] M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *ESORICS*, volume 2808 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2003. (document)

[6] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology-EUROCRYPT 2000, LNCS, Vol. 1807*, pages 259–274, 2000. 2.2

[7] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Proc. CRYPTO 94*, volume 839 of *LNCS*, pages 341–358, 1994. (document)

[8] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545, 2000. 2.4

[9] M. Bellare and P. Rogaway. *Introduction to Modern Cryptography*. 2003. 2.5

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

[11] 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. (document)

[12] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols (the case of encryption-based mutual authentication and key exchange). Cryptology ePrint Archive, Report 2004/334, 2004. (document)

[13] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In Sagiv [28], pages 157–171. (document), 4.3

[14] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. (document), 4.1

[15] EVA. `http://www-eva.imag.fr/`. (document)

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

[17] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 2.3

[18] J. 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. (document)

[19] J. Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, June 2005. (document)

[20] Omer Horvitz and Virgil D. Gligor. Weak key authenticity and the computational completeness of formal encryption. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2003. (document)

[21] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In Sagiv [28], pages 172–185. (document), 2.2, 2.3, 3.2, 3.1, 4.3, 4.3

[22] R. Janvier, Y. Lakhnech, and L. Mazare. (de)compositions of cryptographic schemes and their applications to protocols. Cryptology ePrint Archive, Report 2005/020, 2005. `http://eprint.iacr.org/`. (document)

[23] Y. Lakhnech and L. Mazaré. Computationally Sound Verification of Security Protocols Using Diffie-Hellman Exponentiation. Technical report, Verimag, Centre Équation, 38610 Gières, March 2005. 4.3

[24] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *IEEE Symposium on Security and Privacy*, pages 71–85, 2004. (document)

[25] 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, 2004. (document), 4.3, 4.3

[26] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. *Journal of Computer Security*, 12(1):99–130, 2004. (document)

[27] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*, 2001. (document)

[28] Shmuel Sagiv, editor. *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3444 of *Lecture Notes in Computer Science*, 2005. 13, 21

[29] 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, June 1999. (document)

[30] B. Warinschi. A Computational Analysis of the Needham-Schroeder(-Lowe) protocol. In *Proceedings of 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 248–262, 2003. (document)

A Existence of a N-PAT-SYM-CCA Algorithm

To show that our new criterion makes sens, we prove the existence of a symmetric encryption scheme that is N-PAT-SYM-CCA. However, the algorithm built here is very inefficient as it uses an underlying asymmetric encryption scheme. Let $\mathcal{AE} = (\mathcal{KG}_1, \mathcal{E}_1, \mathcal{D}_1)$ be an asymmetric encryption scheme that is IND-CCA. Let $\mathcal{SS} = (\mathcal{KG}_2, \mathcal{S}, \mathcal{V})$ be a signature scheme secure against UNF. Then the symmetric encryption scheme $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined by: $\mathcal{KG}$ generates a pair of asymmetric keys (pk, sk) using $\mathcal{KG}_1$ as well as a pair of signature keys (sik, vk); the encryption algorithm is defined by $\mathcal{E}(m, (pk, sk, sik, vk)) = m'.\mathcal{S}(m', sik)$ where $m' = \mathcal{E}_1(m, pk)$; and the decryption algorithm verifies that the signature part is valid and decodes the signed message m'.

To prove that $\mathcal{SE}$ is N-PAT-SYM-CCA, it is sufficient to prove that $\mathcal{SE}$ is PAT-SYM-CCA/IND and PAT-SYM-CCA/UNF (this is proven by proposition 3.2).

Let $\mathcal{A}$ be an adversary against PAT-SYM-CCA/UNF. Then it is easy to construct an adversary $\mathcal{A}'$ from $\mathcal{A}$ working against UNF that has the same advantage.

Let $\mathcal{A}$ be an adversary against PAT-SYM-CCA/IND. Then we build an adversary $\mathcal{A}'$ from $\mathcal{A}$ working against IND-CCA such that: $\mathcal{A}'$ is still polynomial; $\mathcal{A}'$ and $\mathcal{A}$ have the same advantage. $\mathcal{A}'$ has to generate a signature key pair and executes $\mathcal{A}$. It uses IND-CCA oracles to simulate its encryption oracle. Note that for the decryption oracle, two cases may occur: m' has not been produced by the IND-CCA encryption oracle, thus the IND-CCA decryption oracle can be used; m' has been produced by the IND-CCA encryption oracle but the signature part is fresh. Then the former adversary (against UNF) can be modified to have the related advantage.

B HASH/IND and HASH/CF imply HASH

Let $\mathcal{H}$ be a hash function that is secure against HASH/IND and HASH/CF. Let us suppose that there exists an adversary $\mathcal{A}$ against HASH/UNF which advantage is not negligible. Then we build the adversary $\mathcal{B}$ against HASH/IND which run $\mathcal{A}$ ($\mathcal{A}$ uses directly oracles given to $\mathcal{B}$).

```
**Adversary** $\mathcal{B}$:
    $pat, bs \leftarrow \mathcal{A}$
    $N' \leftarrow \{0,1\}^\eta$
    $pat' \leftarrow \langle [], N' \rangle$
    $bs' = \underline{\mathcal{H}}^b(pat, pat')$
    **if** $bs = bs'$ **return** $0$
    **else** $b' \leftarrow \{0,1\}$
        **return** $b'$
```

The advantage of $\mathcal{B}$ against IND is detailed thereafter.

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{B}}^{IND} &= pr(\mathcal{B} \rightarrow 0 \text{ in } \mathbf{Exp}_{\mathcal{B}}^{IND} | b = 0) \\
&\quad - pr(\mathcal{B} \rightarrow 0 \text{ in } \mathbf{Exp}_{\mathcal{B}}^{IND} | b = 1) \\
&= pr(\mathbf{Exp}_{\mathcal{A}}^{UNF} = t) + \frac{1}{2}.pr(\mathbf{Exp}_{\mathcal{A}}^{UNF} = f) \\
&\quad - pr(\mathbf{Exp}_{\mathcal{A}'}^{UNF} = t) - \frac{1}{2}.pr(\mathbf{Exp}_{\mathcal{A}'}^{UNF} = f)
\end{aligned}$$

Where $\mathcal{A}'$ is an adversary against UNF defined by:

```
**Adversary** $\mathcal{A}'$:
    $pat, bs \leftarrow \mathcal{A}$
    $N' \leftarrow \{0,1\}^\eta$
    $pat' \leftarrow \langle [], N' \rangle$
    **return** $pat', bs$
```

We obtain

$$2.\mathbf{Adv}_{\mathcal{B}}^{IND} = \mathbf{Adv}_{\mathcal{A}}^{UNF} - \mathbf{Adv}_{\mathcal{A}'}^{UNF}$$

Hence, as $\mathbf{Adv}_{\mathcal{B}}^{IND}$ is negligible and $\mathbf{Adv}_{\mathcal{A}}^{UNF}$ is not, $\mathcal{A}'$ has a non negligible advantage against HASH/UNF.

Finally, we build from $\mathcal{A}$ an adversary $\mathcal{C}$ against collision free which advantage is related to the advantage of $\mathcal{A}'$. For that purpose, $\mathcal{C}$ generates a nonce N^H in order to simulate with a function ρ the hash oracle used by $\mathcal{A}$.

Adversary $\mathcal{C}$:

```
    $N^H \leftarrow \{0,1\}^\eta$
    $pat, bs \leftarrow \mathcal{A}/\underline{k}, \rho$
    $N' \leftarrow \{0,1\}^\eta$
    $N'' \leftarrow \{0,1\}^\eta$
    $pat' \leftarrow \langle [], N' \rangle$
    $pat'' \leftarrow \langle [], N'' \rangle$
    **return** $pat'[N^H], pat''[N^H]$
```

Then, as $PrRand^{CF}$ is negligible, the probability that $\mathcal{C}$ finds a collision is negligible. Moreover, this probability is greater than the probability that $\mathcal{C}$ finds a collision and the hash of $pat'[N^H]$ is equal to the bs produced by $\mathcal{A}$. In the following, events like $\mathcal{H}(pat'[N^H]) = bs$ means: after the random execution of $\mathbf{Exp}_{\mathcal{A}'}^{UNF}$, we obtain pat', N^H and bs such that this equality holds. To deduce the second inequality, we use lemma B.1 that is given later in this appendix.

$$\begin{array}{rcl} pr(\mathbf{Exp}_{\mathcal{C}}^{NC} = t) & \geq & pr\big(\mathcal{H}(pat'[N^H]) = bs = \mathcal{H}(pat''[N^H])\big) \\ & \geq & pr(\mathcal{H}(pat'[N^H]) = bs) \\ & & .pr(\mathcal{H}(pat''[N^H]) = bs) \\ & \geq & \big(pr(\mathbf{Exp}_{\mathcal{A}'}^{UNF} = t)\big)^2 \end{array}$$

There is a contradiction as $\mathcal{A}'$ has a non negligible advantage and $PrRand^{UNF}$ is negligible. Hence $\mathcal{H}$ verifies HASH/UNF.

Lemma B.1 *Let X, Y and Y' be three random variables. X is chosen randomly in a finite set S_X, Y and Y' are chosen randomly in the finite set S_Y. Let E be a predicate over $S_X \times S_Y$. Then*

$$pr\big(E(X,Y) \wedge E(X,Y'\big) \geq \big[pr\big(E(X,Y)\big)\big]^2$$

To prove this lemma, let p be the left probability. Hence,

$$\begin{array}{rcl} p & = & pr\big(E(X,Y) \wedge E(X,Y')\big) \\ & = & \displaystyle\frac{1}{|S_X|}\sum_{x\in S_X} pr\big(E(x,Y) \wedge E(x,Y')\big) \\ & = & \displaystyle\frac{1}{|S_X|}\sum_{x\in S_X} pr\big(E(x,Y)\big).pr\big(E(x,Y')\big) \\ & = & \displaystyle\frac{1}{|S_X|}\sum_{x\in S_X} pr\big[\big(E(x,Y)\big)\big]^2 \end{array}$$

Then, using lemma B.2, we get:

$$\begin{array}{rcl} p & \geq & \displaystyle\frac{1}{|S_X|^2}\sum_{x,x'\in S_X} pr\big[\big(E(x,Y)\big)\big].pr\big[\big(E(x',Y)\big)\big] \\ & \geq & \displaystyle\Big(\frac{1}{|S_X|}\sum_{x\in S_X} pr\big[\big(E(x,Y)\big)\big]\Big)^2 \\ & \geq & \Big(pr\big[\big(E(X,Y)\big)\big]\Big)^2 \end{array}$$

Lemma B.2 *Let $(a_i)_{1\leq i\leq n}$ be n real numbers. Then*

$$\sum_{1\leq i\leq n} a_i^2 \geq \frac{1}{n} \sum_{1\leq i,j\leq n} a_i.a_j$$

By developing $(a_i - a_j)^2 \geq 0$, we obtain

$$a_i^2 + a_j^2 \geq 2.a_i.a_j$$

$$\sum_{1\leq i,j\leq n} a_i^2 + a_j^2 \geq 2. \sum_{1\leq i,j\leq n} a_i.a_j$$

$$2.n. \sum_{1\leq i\leq n} a_i^2 \geq 2. \sum_{1\leq i,j\leq n} a_i.a_j$$

$$\sum_{1\leq i\leq n} a_i^2 \geq \frac{1}{n} \sum_{1\leq i,j\leq n} a_i.a_j$$

C Proof of the Simplified Reduction Theorem

The intuition of the proof relies on the following principle: the adversary $\mathcal{A}^o$ is built using $\mathcal{A}$ as a sub-routine. However, as $\mathcal{A}^o$ works against γ_2, requests made by $\mathcal{A}$ to F_1 are answered using some fresh challenge θ. The adversary $\mathcal{B}$ also uses $\mathcal{A}$ as a sub-routine and works against γ_1. It is designed in such a way that whenever the challenge bit b of γ_1 equals 1, the experiment involving $\mathcal{B}$ against γ_1 is similar to the experiment involving $\mathcal{A}$ against γ. When b equals 0 then the experiment involving $\mathcal{B}$ against γ_1 is similar to the experiment involving $\mathcal{A}^o$ against γ_2.

The formal definitions for adversaries $\mathcal{A}^o$ and $\mathcal{B}$ are detailed thereafter:

```
Adversary $\mathcal{A}^o$:
    $\theta_1 \leftarrow \Theta_1(\eta)$
    $\theta'_2 \leftarrow \Theta_2(\eta)$
    $s \leftarrow \mathcal{A}/\eta, \lambda s.F_1(s,\theta_1,\theta'_2), \underline{F_2}$
    return $s$
```

```
Adversary $\mathcal{B}$:
    $\theta_2 \leftarrow \Theta_2(\eta)$
    $\theta'_2 \leftarrow \Theta_2(\eta)$
    $s \leftarrow \mathcal{A}/\eta, \lambda s.\underline{G}(H(s,\theta_2,\theta'_2)), \lambda s.F_2(s,\theta_2)$
    if $V_2(s,\theta_2)$ return $1$
    else return $0$
```

It is now possible to relate the advantages of our three different adversaries. For that purpose, note that the experiment involving $\mathcal{B}$ is successful in two cases:

- If $b = 1$ and $\mathcal{B}$ outputs 1. Then the experiment $\mathbf{Exp}_{\mathcal{B}}^{\gamma_1}$ is similar to $\mathbf{Exp}_{\mathcal{A}}^{\gamma}$ To prove this, we detail the two experiments:

```
Experiment $\mathbf{Exp}_{\mathcal{B}}^{\gamma_1}(\eta)$:
    $b \leftarrow [0,1]$
    $\theta_1 \leftarrow \Theta(\eta)$
    $\theta_2 \leftarrow \Theta_2(\eta)$
    $\theta'_2 \leftarrow \Theta_2(\eta)$
    $s \leftarrow \mathcal{A}/\eta, \lambda s.G(H(s,\theta_2,\theta'_2),b,\theta_1),$
            $\lambda s.F_2(s,\theta_2)$
    if $V_2(s,\theta_2)$ return $verif_b(1)$
    else return $verif_b(0)$
```

```
Experiment $\mathbf{Exp}_{\mathcal{A}}^{\gamma}(\eta)$:
    $\theta_1 \leftarrow \Theta_1(\eta)$
    $\theta_2 \leftarrow \Theta_2(\eta)$
    $s \leftarrow \mathcal{A}/\eta, \lambda s.F_1(s,\theta_1,\theta_2),$
            $\lambda s.F_2(s,\theta_2)$
    return $V_2(s,\theta_2)$
```

Using the assumption on G, $G(H(s,\theta_2,\theta'_2),1,\theta_1) = F_1(s,\theta_1,\theta_2)$ the equivalence of the two experiments appears clearly in the case $b = 1$. Moreover, $\mathcal{B}$ outputs 1 means that $\mathcal{A}$ solved its challenge.

- In a similar way, if $b = 0$ then the experiment $\mathbf{Exp}^{\gamma_1}_{\mathcal{B}}$ is similar to $\mathbf{Exp}^{\gamma_2}_{\mathcal{A}^o}$ (their boolean outputs are opposite), $\mathcal{B}$ outputs 0 means that $\mathcal{A}^o$ failed to solve the challenge. Using the assumption on G, $G(H(s, \theta_2, \theta_2'), 0, \theta_1) = F_1(s, \theta_1, \theta_2')$ the equivalence between the two experiment is immediate.

  Experiment $\mathbf{Exp}^{\gamma_1}_{\mathcal{B}}(\eta)$:

  ```
      $b \leftarrow [0, 1]$
      $\theta_1 \leftarrow \Theta(\eta)$
      $\theta_2 \leftarrow \Theta_2(\eta)$
      $\theta_2' \leftarrow \Theta_2(\eta)$
      $s \leftarrow \mathcal{A}/\eta, \lambda s.G(H(s, \theta_2, \theta_2'), b, \theta_1),$
              $\lambda s.F_2(s, \theta_2)$
      **if** $V_2(s, \theta_2)$ **return** $verif_b(1)$
      **else return** $verif_b(0)$
  ```

  Experiment $\mathbf{Exp}^{\gamma_2}_{\mathcal{A}^0}(\eta)$:

  ```
      $\theta_2 \leftarrow \Theta_2(\eta)$
      $\theta_1 \leftarrow \Theta_1(\eta)$
      $\theta_2' \leftarrow \Theta_2(\eta)$
      $s \leftarrow \mathcal{A}/\eta, \lambda s.F_1(s, \theta_1, \theta_2'),$
              $\lambda s.F_2(s, \theta_2)$
      **return** $V_2(s, \theta_2)$
  ```

$$
\begin{aligned}
\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta) &= 2.\big(Pr[\mathbf{Exp}^{\gamma_1}_{\mathcal{B}}(\eta) = true] - PrRand^{\gamma_1}\big) \\
&= Pr[\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta) = true] \\
&\quad + Pr[\mathbf{Exp}^{\gamma_2}_{\mathcal{A}^o}(\eta) = false] - 1 \\
&= Pr[\mathbf{Exp}^{\gamma}_{\mathcal{A}}(\eta) = true] - PrRand^{\gamma} \\
&\quad + PrRand^{\gamma_2} - Pr[\mathbf{Exp}^{\gamma_2}_{\mathcal{A}^o}(\eta) = true] \\
&= \frac{1}{2}\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) - \frac{1}{2}\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)
\end{aligned}
$$

In this computation, we used that $PrRand^{\gamma_1} = 1/2$ as bit b is chosen among two possible values. We also used that $PrRand^{\gamma} = PrRand^{\gamma_2}$ which is true because γ and γ_2 have the same verification oracle V_2.

This gives the awaited result:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)|$$

Proof of the Reduction Theorem

Adversary $\mathcal{A}^1$ represents adversary $\mathcal{A}$ trying to solve its challenge against V_1.

Adversary $\mathcal{A}^1$:

```
    $\theta_2 \leftarrow \Theta_2(\eta)$
    $s \leftarrow \mathcal{A}/\eta, \lambda s.\underline{G}(H(s, \theta_2, \theta_2)), \lambda s.F_2(s, \theta_2)$
    **return** $s$
```

PRTM $\mathcal{A}$ can gain its advantage by solving challenge V_1 or challenge V_2. As we suppose that a string can solve at most one challenge, the following equality holds where γ, V_i denotes criterion γ using only V_i as verifier.

$$\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) = \mathbf{Adv}^{\gamma,V_1}_{\mathcal{A}}(\eta) + \mathbf{Adv}^{\gamma,V_2}_{\mathcal{A}}(\eta)$$

Then, by keeping the same construction as above, the advantage against V_2 is known. Moreover, the advantage of $\mathcal{A}$ against V_1 is equal to the advantage of $\mathcal{A}^1$ against γ_3.

$$\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) = \mathbf{Adv}^{\gamma_3}_{\mathcal{A}^1}(\eta) + \mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o} + 2.\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)$$

This gives the conclusion of the theorem:

$$|\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta)| \leq 2.|\mathbf{Adv}^{\gamma_1}_{\mathcal{B}}(\eta)| + |\mathbf{Adv}^{\gamma_2}_{\mathcal{A}^o}(\eta)| + |\mathbf{Adv}^{\gamma_3}_{\mathcal{A}^1}(\eta)|$$

D Application of the Reduction Theorems

D.1 From N-PAT-IND-CCA to IND-CCA

In order to reduce the N-PAT-IND-CCA criterion (denoted by γ_N), we only need the simplified version of the reduction theorem. In N-PAT-IND-CCA, encoded messages can be patterns and there is an order among keys: sk_i can be encoded using pk_j iff $i > j$. The reduction operates from γ_{N+1} to γ_N and γ (i.e. IND-CCA) as follows.

- Θ_1 generates the key pair (pk_1, sk_1).
- Θ_2 generates the other key pairs (pk_2, sk_2) to (pk_{N+1}, sk_{N+1}) as well as the challenge bit b.
- F_1 (resp. F_2) is the oracle for encryption, decryption, public key related to key pairs in θ_1 (resp. in θ_2).
- V_2 verifies that bit b has been correctly guessed.
- H is the identity when considering decryption and public key emission and G is exactly F_1 in that case.
- G is the classical left-right encryption and $H(s, \theta_2, \theta_2')$ is defined as follows:

  $$H(\langle pat_0, pat_1\rangle, \theta_2, \theta_2') = \langle v(pat_{b_2'}, \theta_2'), v(pat_{b_2}, \theta_2)\rangle$$

  Where b_2 (resp. b_2') is the challenge bit contained in θ_2 (resp. θ_2').

We first want to verify that (γ, γ_N) defines a valid simplified partition of γ_{N+1}.

- As secret key sk_1 cannot occur under any public key, F_2 only depends on θ_2.
- Verifier V_2 only depends on θ_2.

As (γ, γ_N) is a valid simplified partition of γ_{N+1}, it is possible to apply the simplified version of the reduction theorem. For any PRTM $\mathcal{A}$, there exist two PRTM $\mathcal{B}$ and $\mathcal{A}^o$ such that:

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

It is then possible to conclude using a simple recursion.

D.2 From N-PAT-SYM-CCA to SYM-CCA/IND and SYM-CCA/UNF

In order to reduce the N-PAT-SYM-CCA criterion (denoted by γ_N), we need the full version of the reduction theorem. As in N-PAT-IND-CCA, encoded messages can be patterns and there is an order among keys: k_i can be encoded using k_j iff $i > j$, but there are also two ways to win the challenge, either by guessing the value of bit b (criterion SYM-CCA/IND) or by forging an encoded message without using the encryption oracles (criterion SYM-CCA/UNF).

The reduction operates from γ_{N+1} to γ_N, γ_{IND} (i.e. SYM-CCA/IND) and γ_{UNF} (i.e. SYM-CCA/UNF) as follows.

- Θ_1 generates key k_1.
- Θ_2 generates the other keys k_2 to k_{N+1} as well as the challenge bit b.
- F_1 (resp. F_2) is the oracle for encryption and decryption related to key(s) in θ_1 (resp. in θ_2).
- V_2 verifies that bit b has been correctly guessed or that the final output is an encoded message by a key from θ_2 that has not been produced by an encryption oracle.
- V_1 verifies that the output message is encoded by k_1 and has not been produced by F_1.
- H is the identity when considering decryption and G is exactly F_1 in that case.

- G is the classical left-right encryption and $H(s, \theta_2, \theta'_2)$ is defined as follows:

$$H(\langle pat_0, pat_1\rangle, \theta_2, \theta'_2) = \langle v(pat_{b'_2}, \theta'_2), v(pat_{b_2}, \theta_2)\rangle$$

  Where b_2 (resp. b'_2) is the challenge bit contained in θ_2 (resp. θ'_2).

We first want to verify that $(\gamma_{IND}, \gamma_N, \gamma_{UNF})$ defines a valid partition of γ_{N+1}.

- As key k_1 cannot occur under any public key, F_2 only depends on θ_2.
- Verifier V_2 only depends on θ_2 and V_1 only depends on θ_1

Partition $(\gamma_{IND}, \gamma_N, \gamma_{UNF})$ is a valid partition of γ_{N+1}, it is possible to apply the reduction theorem. For any PRTM $\mathcal{A}$, there exist three PRTM $\mathcal{B}$, $\mathcal{A}^o$ and $\mathcal{A}^1$ such that:

$$|\mathbf{Adv}_{\mathcal{A}}^{\gamma_{N+1}}| \leq 2.|\mathbf{Adv}_{\mathcal{B}}^{\gamma_{IND}}| + |\mathbf{Adv}_{\mathcal{A}^o}^{\gamma_N}| + |\mathbf{Adv}_{\mathcal{A}^1}^{\gamma_{UNF}}|$$

It is then possible to conclude using a simple recursion.

D.3 Mixing all Criteria

Let us define the N-PAT-ASYM-SYM-SIGN-HASH-CCA (N-PASSH-CCA) criterion as $\gamma = (\Theta, F, V)$ where Θ is composed of four parts:

- Θ_a generates N pairs of asymmetric keys (pk_1, sk_1) to (pk_N, sk_N).
- Θ_b generates N symmetric keys k_1 to k_N.
- Θ_c generates N pairs of signature keys (sik_1, vk_1) to (sik_N, vk_N).
- Θ_d generates a nonce N^H, a key k as well as a challenge bit b.

F is also split in four parts:

- F_a corresponds to the oracles using θ_a as in N-PAT-IND-CCA except that patterns can also ask for symmetric encryption, symmetric keys, signature of a message, signature keys, hashing of a message and nonce N^H. F_a depends on θ_a, θ_b, θ_c and θ_d.
- F_b corresponds to oracles using θ_b as in N-PAT-SYM-CCA, patterns are also extended but cannot include asymmetric keys from θ_a. F_b depends on θ_b, θ_c and θ_d.
- F_c corresponds to oracles using θ_c as in N-UNF, F_c depends only on θ_c.
- F_d corresponds to oracles using θ_d as in HASH, F_d depends only on θ_c.

Finally V is also a disjunction of five parts:

- V_{IND} answers true if its argument if the bit b in Θ_d.
- $V_{UNF-SYM}$ answers true if it receives a symmetric encryption not forged by F_b.
- $V_{UNF-SIGN}$ answers true if it receives a signature not forged by F_c.
- $V_{UNF-HASH}$ answers true if it receives a pair h, pat where $h = \mathcal{H}(k, v(pat, N^H)$ and h has not been forged using F_d.
- $V_{CF-HASH}$ answers true if it receives a pair of distinct bit-strings bs_0, bs_1 that have the same hash.

E Proof of Theorem 4.1

The intuition is that if an adversary $\mathcal{A}$ can produce a NDY trace, then it is able to break one of the cryptographic schemes. Let Q be the number of atoms (keys and nonces) that occur in $\mathcal{A}$. We build a Q-PASSH-CCA (criterion denoted by γ) adversary $\mathcal{B}$ such that if p is the probability:

$$p = Pr\big[t_f, t_c \leftarrow Exec(\mathcal{A}, \Pi) \text{ and } t_f \notin traces(\Pi)\big]$$

We have the following majoration of p.

$$p \leq \big(2.Q + 7\big).\mathbf{Adv}^{\gamma}_{\mathcal{B}}(\eta) + f(\eta) \tag{3}$$

where f is a negligible function. Using proposition 3.3, it is possible to deduce that the probability for $\mathcal{A}$ to produce a non Dolev-Yao trace is negligible.

Our Q-PASSH-CCA adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine and deduces a string solving its challenge (for example the challenge bit b or a new signature) as soon as $Exec(\mathcal{A}, \Pi)$ produces an invalid trace. Using its own oracles, $\mathcal{B}$ simulates $Exec(\mathcal{A}, \Pi)$ and produces the formal trace in order to find a non-deducible (NDY) message.

During its initialization, the adversary $\mathcal{B}$ randomly chooses an integer i between 0 and Q. If $i \neq 0$, then the i^{th} nonce generated by $\mathcal{B}$ (denoted by N) is trapped. In order to answer queries from $\mathcal{A}$, $\mathcal{B}$ randomly generates identities and nonces from Π except N. $\mathcal{B}$ uses its challenge keys for the different keys in Π. For nonce N, $\mathcal{B}$ generates two nonces N_0 and N_1, $\mathcal{B}$ uses its oracles in such a way that messages involving N uses $N_0.N^H$ (resp. $N_1.N^H$) when the challenge bit b is 0 (resp. 1). N^H is the challenge nonce related to hashing in PASSH-CCA (as $\mathcal{B}$ does not know if N_0 or N_1 is used, this is required in order to compute the hashing of a message involving N using an oracle).

When $\mathcal{A}$ waits for a message m, $\mathcal{B}$ has to forge $m = \langle m_1, ..., m_n\rangle$ where messages m_i are not pairs of messages. Then $\mathcal{B}$ generates each m_i using its oracles (e.g. if m_i is an encoding using pk, $\mathcal{B}$ uses the left-right encryption oracle related to pk). If N appears ”under” a left-right oracle, then $N_0.[N^H]$ (resp. $N_1.[N^H]$) is used for the left (resp. right) argument of the oracle. If N appears anywhere else it is impossible for $\mathcal{B}$ to continue the protocol simulation. Hence $\mathcal{B}$ aborts its execution. Note that $\mathcal{B}$ cannot be asked to reveal a secret key, a signature key or a symmetric key in a message m_i (such keys have to be protected by an encryption layer and so a left-right oracle is used with a pattern asking for the key).

When $\mathcal{A}$ emits a message m, m is parsed according to the protocol specification. 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 its decryption oracles. To achieve parsing, $\mathcal{B}$ has to be able to test whether a string is a secret/signature/symmetric key, this can easily be achieved using oracles.

Eventually, $\mathcal{A}$ stops. Then $\mathcal{B}$ checks that there are no collisions between two messages parsed as hash. If this is not the case, $\mathcal{B}$ wins against HASH/CF, this event is denoted by E_0. Else if the trace is NDY then $\mathcal{B}$ knows the first NDY message m and a recursive procedure is applied on m in order to win the challenge.

1. If m is $N_0.N^H$ or $N_1.N^H$, $\mathcal{B}$ deduces the challenge bit b.
2. If m is another nonce, $\mathcal{B}$ aborts.
3. If m is a secret key or a symmetric key, $\mathcal{B}$ also deduces b.
4. If m is a signature key, $\mathcal{B}$ can forge a new fresh signature and thus wins its challenge.
5. If m is a pair $\langle m_1, m_2\rangle$, then m_1 or m_2 is NDY and this procedure is applied recursively.
6. If m is an asymmetric encryption $\{m'\}_{pk}$, as m is NDY it has not been produced by an oracle (otherwise, m would have circulated not protected). Hence using the decryption oracle, $\mathcal{B}$ obtains m' which is also NDY.
7. If m is a signature or a symmetric encoding, m is NDY thus it has not been produced by an oracle and $\mathcal{B}$ has forged a new signature or a new symmetric encoding.
8. If m is a hashing $h(m')$, then m' has to be known (to test m during the protocol execution). If m' contains N, then $\mathcal{B}$ can deduce a hollow pattern pat such that $\mathcal{H}(k, v(pat, N^H)) = h$. Hence $\mathcal{B}$ wins. Else, $\mathcal{B}$ aborts.

9. If m is a hashing $h(m')$ and m' does not contain N, then $\mathcal{B}$ aborts.

Whenever $\mathcal{B}$ decides to abort, it answers a random bit for the challenge bit b.

If $\mathcal{A}$ produces an invalid trace, then we consider the different answers that the former procedure can have produced. E_i denotes the event where the procedure stopped in the i^{th} case of the list. Hence,

$$p = \sum_{i=0}^{9} Pr(E_i)$$

As nonce N is chosen randomly, $Pr(E_2)$ and $Pr(E_9)$ are lower than respectively $Q.Pr(E_1)$ and $Q.Pr(E_8)$. Moreover, events E_i for i different from 2, 5 and 9 imply that $\mathcal{B}$ wins its challenge without aborting. Let us call B (resp. $\neg B$) the event where $\mathcal{B}$ does not abort (resp. aborts). Hence,

$$p \leq \big(2.Q + 7\big).Pr(B)$$

As $PrRand$ is negligible for criteria related to UNF, there exists a negligible function g such that:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{B}}^{\gamma}(\eta) &= 2.Pr(\mathcal{B}wins) - 1 - g(\eta)\\ &= 2.Pr(B) + Pr(\neg B) - 1 - g(\eta)\\ &= Pr(B) - g(\eta)\end{aligned}$$

Hence, it is easy to obtain formula 3 and the awaited result.

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.