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

R. Janvier, Y. Lakhnech, L. Mazaré
VERIMAG
2, av. de Vignates, 38610 Gières - FRANCE
{romain.janvier,yassine.lakhnech,laurent.mazare}@imag.fr

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. We go further and consider equational theories and Cipher Block Chaining. We show how to map certain type of equational theories into the computational world and prove correctness of an extended Dolev-Yao model that deals with Cipher Block Chaining.

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.

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

Introduction

This paper is about relating the so-called formal [1] and computational models for cryptographic protocols. We consider protocols that may combine asymmetric and symmetric encryption, signature and hashing and show that under reasonable cryptographic hypothesis, a secrecy property established in the formal model is also true in the computational.

Motivation Historically, verification of cryptographic protocols has been separated in two distinct branches. *Formal verification* of cryptographic protocols, originates from the work of Dolev and Yao [12]. The essential part of this approach is the perfect cryptography assumption that can be roughly summarized as follows: messages are represented as algebraic terms, fresh nonce creation is perfect, that is, nonces range over an infinite domain and freshness is absolute, the same holds for key creation. Moreover, there is no way to guess a nonce or a key and 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. Despite the strong assumptions concerning the cryptographic primitives, flaws have been found on 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 [10]). The good news about this approach is that a rich collection of automatic verification methods and tools have been developed [20, 8, 21, 15, 7, 22].

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 [13, 4] of any attacker. Protocols as well as attackers are randomized polynomial-time Turing machines. This computational approach is recognized as more realistic than the formal 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 schema S_i is correct with respect to the security notion C_i and if some additional syntactic conditions are satisfied by Π then the formal model is a safe abstraction of the computational, that is, correctness of the protocol established in the formal model implies its correctness in the computational one.

are of extreme importance for gaining confidence that a cryptographic protocol is secure.

[1]Formal does not here mean rigorous but should be understood in the sense of formal languages

Contributions and related work 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. The first work to mention here is maybe Abadi and Rogaway's paper [1]. They proved that a notion of message *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 equivalents cannot be separated by a Turing machine in a reasonable (polynomial) time. This paper deals, however, with passive attackers that do not intervene during protocol execution. Active attackers are considered in [23, 18, 2, 16, 11, 19]. Essentially, these papers prove that if the encryption scheme verifies a certain property (e.g. IND-CCA), then security in the formal model implies security in the computational model. However, each of this papers lacks generality. For instance the results in [23, 18] does not deal with secret key transmission. Cortier and Warinschi prove in [11] safety of the formal model for protocols that use asymmetric encryption and signature under the assumption that secret and signature keys are not sent. A similar result is proved by Backes, Pflizmann and Waidner [2]. In the latter, the formal model is not exactly the Dolev-Yao model, although very close. In fact, it can be seen as a kind of implementation of Dolev-Yao's model. The principal drawback is that it is not clear whether properties in this model can be checked automatically whereas verification in the Dolev-Yao model can be achieved efficiently. P. Laud [16] proves safety of the formal model for symmetric encryption. In particular, he deals with encryption cycles.

In conclusion, none of these papers deals with hashing and protocols that combine asymmetric and symmetric encryption, signature and hashing. The main result of our paper does this.

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 $C1$ 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 and non-collision.

Then, we exploit our security criterion and reduction theorem to establish safety of the formal model for protocols that may combine asymmetric and symmetric cryptography, message signing and hash functions. Safety here means that a cryptographic protocol that is proven correct in the formal model is also correct in the computational one. Moreover, we extend this result for cryptographic schemes that are based on Cipher Block Chaining or have some algebraic properties. The practical consequences of these results are obvious. Automatic verification of security protocols based on the formal model can be safely applied.

Paper organization The next section gives the necessary definitions for using both the computational model and the formal 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 two reduction theorems and their proofs. Then, this theorem is applied in section 4 to reduce all our criteria. Section 5 uses these results to show that, under some quite unrestrictive hypothesis, the formal model is a safe abstraction of the computational model. The objective of section 6 is to weaken the IND-CCA hypothesis and to show that the abstraction is still safe if we consider the formal model with equational theories and the computational model with a modified IND-CCA criterion. Finally, a quick conclusion of this paper is 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 cypher-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 polynomially bounded by η.

A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined by three algorithms. The key generation algorithm

$\mathcal{KG}$ is a randomized function which given a security parameter η outputs a key k. The encryption algorithm $\mathcal{E}$ is also a randomized function which given a message and a key outputs the encryption of the message by this key. Finally the decryption algorithm $\mathcal{D}$ takes as input a key and a cypher-text and outputs the corresponding plain-text, i.e., $\mathcal{D}(\mathcal{E}(m, k), k) = m$. The execution time of the three algorithms is also 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 $\mathcal{KG}$. 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 computes from a key k and a bit-string bs 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 which 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 $\mathcal{A}/F$ are 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 $x > N_c$.

1.3 Definitions for the Formal Model

In this section, we give the basic definitions that are used to introduce the formal [2] 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 $Nonces$ 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. 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 is written $\langle m, n\rangle$. Encryption of message m with key k is denoted by $\{m\}_k$. 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 [12]) 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 $E \vdash \langle m, n\rangle$, 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 $E \vdash m$, then $E \vdash h(m)$.

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.

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

[2] Formal is not used here in the sense of rigorous but denotes the use of formal methods.

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 key (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 comprehension, 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 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 [19] where more discussion is available. In the classical N-IND-CCA criterion (see [3] about N-IND-CCA and its reduction to IND-CCA), the adversary can only use the left-right oracles with messages that it can compute. Since it has no information concerning secret keys, it 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.

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

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 $\cdot$.

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

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 a left-right pattern encryption oracle 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. To win, the adversary has to guess the value of secret bit b.

Criterion N-PAT-IND-CCA is denoted by $\gamma_N = (\Theta; F; V)$, where Θ generates N pairs of keys using $\mathcal{KG}$ and a bit b at random; 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.

An asymmetric encryption scheme $\mathcal{AE}$ is said N-PAT-IND-CCA iff for any adversary $\mathcal{A}$ in $PRTM$, $\mathbf{Adv}^{\gamma_N}_{\mathcal{AE},\mathcal{A}}(\eta)$ 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 adaptation of Selective Forgery Against Adaptive Chosen Message Attack (a good survey on properties for signature schemes is available in [14]). It was also already defined in [19]. 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 $\mathcal{S}_{sik_i}(.)$.

Formally, the N-UNF criterion is $\gamma_N = (\Theta, F, V)$ 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, signature scheme $\mathcal{SS}$ is said N-UNF iff for any adversary $\mathcal{A}$, $\mathbf{Adv}^{N-UNF}_{\mathcal{SS},\mathcal{A}}(\eta)$ is negligible. When $N = 1$, N-UNF can be written UNF.

2.4 The N-PAT-SYM-CCA Criterion

In some sense a symmetric encryption scheme includes both aspects indistinguishability and authentication that are present in asymmetric encryption and message signature respectively [5]. Therefore, our criterion for symmetric encryption is in 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. 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 hold: 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 $PRTM$, $\mathbf{Adv}^{\gamma_N}_{\mathcal{SE},\mathcal{A}}(\eta)$ is negligible.

Existence of a SYM-CCA encryption scheme can be proved under the assumption that there exists an IND-CCA asymmetric encryption scheme and an UNF signature scheme. This is detailed in appendix A.

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 has the same hash for a given key. More details about criteria related to HASH can be found in [6].

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. ■

It is not clear to us if there exists an algorithm that verifies $HASH$.

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 break a criterion into simpler ones.

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:

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

  Oracle G operates on a string of bits, thus it must receive two challenge informations, 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*

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

To extend this result, we consider the case where the verification algorithm V does not only depend on θ_2 (like V_2 before) but also uses θ_1. Hence, the PRTM V is represented by $V_1 \vee V_2$ where V_1 (resp. V_2) depends only on θ_1 (resp. θ_2). V returns true if V_1 or V_2 returns true. γ_1 and γ_2 are defined as above but a new criterion $\gamma_3 = (b, \theta_1; G; V_1)$ occurs in the partition. Then $(\gamma_1, \gamma_2, \gamma_3)$ is a valid partition of γ. However, we add the hypothesis that 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 it 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*

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

3.2 Proof of the Simplified Reduction Theorem

The intuition of the proof relies on the following principle: if none of the queries made by $\mathcal{A}$ to oracle F_1 are answered correctly, then $\mathcal{A}$ must have *a priori* less advantage than if its queries are answered correctly.

We construct the adversary $\mathcal{A}^o$ against the criterion $\gamma_2 = (\theta_2, F_2, V_2)$ which simulates $\mathcal{A}$ and thus has to answer to the queries made to oracle F_1. Since $\mathcal{A}^o$ cannot construct correctly this oracle (as it does not have access to θ_2), 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 easily be reduced to the criterion γ_2.

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 $s$
```

If 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 $\gamma_1 = (b, \theta_1, G, Verif_b)$. It generates a challenge for $\mathcal{A}$. Then 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 $b = 0$.

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 PRTM. For that purpose, note that the experiment involving $\mathcal{B}$ is successful in two cases: if $b = 1$, then $\mathcal{A}$ is confronted to its real oracle and $\mathcal{B}$ outputs 1 means that $\mathcal{A}$ solved its challenge. If $b = 0$, then $\mathcal{A}$ uses oracles as in $\mathcal{A}^o$ and $\mathcal{B}$ outputs 0 means that $\mathcal{A}^o$ failed to solve the challenge.

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

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:

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

3.3 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:

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

4 Applications of the Reduction Theorems

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

4.1 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 long 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}, thus 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:

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

It is then possible to conclude using a simple recursion.

Proposition 4.1 *If an encryption scheme is secure against IND-CCA, then it is secure against N-IND-CCA for any N.*

4.2 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 long 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}, thus 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.

Proposition 4.2 *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.*

4.3 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 long 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.

Proposition 4.3 *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.*

For space reason, we only present here the first step of the proof, the following 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 SIG-CCA 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.

4.4 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 4.4 *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 which 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 which 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 which PRTM C_i 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 4.5 *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.

5 Dolev-Yao is a Safe Abstraction

In the following, we use the classical definitions recalled in [19] for protocols (usually denoted by Π) and the adversary model.

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

Hypothesis over Cryptographic Schemes

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 SYM-CCA and $PrRand^{SYM/UNF}$ is negligible.
- The signature scheme $\mathcal{SS}$ is UNF and $PrRand^{UNF}$ is negligible.
- The hashing algorithm $\mathcal{H}$ is HASH and $PrRand^{HASH/UNF}$ and $PrRand^{HASH/CF}$ are negligible.

Hypothesis over Protocols

There are also a few restrictions over protocols Π considered. These restrictions are defined in the formal world (as they are easier to check there with automated tools).

- Π has to be executable.

- 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 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 honnest agents contains a nonce that remains secret.

5.1 Non Dolev-Yao Traces

Definition 5.1 (Traces) *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. Message m_1 and m_2 can be "-" when no message is received or sent. Assignment $t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c$ denotes that the trace t_c is obtained by the computational Adversary $\mathcal{A}_c$ confronted to Π_c. We assume that only messages accepted by an agent appear in the trace.*

We define a parsing algorithm α that transforms any computational trace related to a given protocol to a formal trace. This algorithm is mainly defined as in [19] except that it is extended for hashing. The operation can be seen as a verification made by all the honest agents working together. They take each concrete message of the trace and replace it by a Dolev-Yao term. That is, all the bit-strings corresponding to atoms are associated to Dolev-Yao atoms such as $N_A, H_1, H_2, I_1, Pk_{H_1}, \ldots$. Bit-strings corresponding to concatenation of messages are associated to $\langle T_1, T_2 \rangle$, encryptions are associated to $\{T\}_K$ and hash are associated to $h(T)$. Messages are parsed according to the protocol, i.e. only the decompositions necessary for the execution of the protocol are made. When a bit-string cannot be tested (e.g. a nonce generated by the adversary, a hash tested later) it is assumed to be a fresh nonce generated by the adversary. If later, this string is parsed otherwise (e.g. as a hash), the nonce is replaced by the new term in all the previous messages.

To illustrate our parsing algorithm, let us consider the following example. The objective is to illustrate how some information learned at the end of the trace can be used to modify the beginning of the trace. Our simple protocol is defined by:

$$\begin{array}{rcl} A \rightarrow B & : & h(N_A) \\ A \rightarrow B & : & N_A \end{array}$$

Let us consider the computational trace in which the intruder sends the hashing of a nonce N to B and then sends the nonce N itself to B. The parsing algorithm operates as follows.

The first received message is a hash message, hence it is impossible to obtain its content. This first message is added in the trace as $(X, -, B, s)$. When the second message is received, the algorithm knows every information present in the former hash message. Hence, it tests if the hash is valid and as it is, X is replaced by $h(X')$. The trace is now $h(X'), -, B, s); (X', -, B, s)$. As the computational trace is finished, the algorithm replaces every variables using fresh nonces. Hence the final formal trace is:

$$\big(h(N_I), -, B, s\big); \big(N_I, -, B, s\big)$$

Definition 5.2 (Non Dolev-Yao Traces) *A formal trace t_f is said* Non Dolev-Yao *(NDY) if 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 if $\alpha(t_c)$ is NDY or if there is a hash message that corresponds to hashing of two distinct messages.*

5.2 Relating Computational and Formal Traces

In this section, we formulate the main theorem. It states that if the condition 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 [18] 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 [19]. 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.

Theorem 5.1 *Let Π be a protocol verifying the conditions given above. Then for any concrete Adversary $\mathcal{A}_c$:*

$$Pr\big[t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c \ ; \ t_c \ NDY\big] \textit{ is negligible}$$

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

Now, it is possible to relate formal and computational properties Let P_c denote a property in the computational world represented by a predicate over computational traces.

Hence, a protocol Π verifies P_c (denoted by $\Pi \models_c P_c$) iff for any Adversary $\mathcal{A}_c$,

$$Pr\big[t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c \ ; \ \neg P_c(t_c)\big] \text{ is negligible}$$

Let P_f denote a property in the formal world represented by a predicate 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.

There is an intuitive relation between formal and computational properties which is expressed in the following proposition: proving formally P_f allows us to deduce P_c.

Proposition 5.1 *Let P_f and P_c be a formal and a computational property such that*

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

If the conditions given above still hold, then

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

This proposition states that if the formal property correctly under-approximates the computational property then the formal abstraction is correct. It has been applied to mutual authentification in [18] in which there is also a longer discussion about formal/computational properties. We applied it to secrecy as a trace property in [19]. Another version of secrecy (given as the probability to get any information on the secret) appears in [11].

6 Extending the Dolev-Yao Model

The objective of this section is to weaken the perfect cryptography hypothesis. This is first done in the formal model by adding a new rule to possible deductions. Then in the computational model, the classical IND-CCA criterion has to be modified. Hence, a soundness theorem can show that the extended Dolev-Yao model is still a safe abstraction of the computational model if we suppose that the encryption scheme verifies the modified IND-CCA property.

To enhance the intruder capacity in both models, let us consider a pair of functions: δ_f takes a message as argument and outputs another message and δ_c uses a string of bits to produce another string of bits. Furthermore, these two functions have to be related: they must modify their arguments in the same way.

Definition 6.1 (DY-weakening) *A DY-weakening is a pair of function (δ_f, δ_c) such that:*

$$\alpha(\delta_c(s)) = \delta_f(\alpha(s))$$

Where α is the parsing algorithm (we suppose that in the two occurrences of this algorithm, the same names are associated to the same nonces exactly). We also ask that given a string s, the set $S' = \{\delta_c^n(s), n > 0\}$ can be computed in polynomial time.

This last hypothesis is useful when defining the computational criterion.

To illustrate this definition, we give a classical example: Cipher Block Chaining. CBC allow the intruder to make some deduction using a property verified on some block encryption schemes.

Example 6.1 (Cipher Block Chaining) *CBC is a DY-weakening that allows the intruder to deduce from an encoding of message m the encoding of a prefix of m. More formally,*

$$\begin{array}{rcl} \delta_f(\{\langle m, n\rangle\}_{pk}) & = & \{m\}_{pk} \\ \delta_c(\mathcal{E}(s.s', pk)) & = & \mathcal{E}(s, pk) \end{array}$$

It is easy to verify that (δ_f, δ_c) constitutes a correct DY-weakening. Proof of the existence of a δ_c-IND-CCA encryption scheme for CBC is given in appendix D.

6.1 Extended Dolev-Yao Model

In the DY model, the deductibility relation $T \vdash m$ is defined by inference rules. We add a new inference that represents that an intruder can deduce from a message its image after applying δ_f.

$$\frac{T \vdash m}{T \vdash \delta_f(m)}$$

This defines an Extended Dolev-Yao model denoted by EDY. Note that usually, δ_f is defined on a subset of the set of messages. If δ_f cannot be applied to a message m, then our inference add no new deductions starting from m.

Example 6.2 (Cipher Block Chaining) *Using δ_f given in the previous example, we obtain the classical prefix rule:*

$$\frac{T \vdash \{\langle m, n\rangle\}_{pk}}{T \vdash \{m\}_{pk}}$$

6.2 The δ_c-IND-CCA Criterion

If we consider an IND-CCA encryption scheme, then EDY is still a safe abstraction of the computational model as EDY is an extension of DY (formally, $T \vdash_{DY} m$ implies $T \vdash_{EDY} m$). However, EDY can still be a safe abstraction when considering an encryption scheme that is not IND-CCA. The encryption scheme has to be secure against a new criterion that uses δ_c, this criterion is called δ_c-IND-CCA.

δ_c-IND-CCA is defined as a criterion $(\mathcal{KG}, \mathcal{E}_{pk}, \mathcal{D}_{sk})$ similar to IND-CCA. The only difference is that in IND-CCA the decryption oracle does not apply to strings produced by the encryption oracle (stored in a set of strings S), here it does not apply to strings that are in $S' = \{\delta_c^n(s), s \in$

$S, n > 0\big\}$. As this set can be computed in polynomial time, the decryption oracle can be achieved using a PRTM.

It is easy to extend δ_c-IND-CCA to $N - \delta_c$-IND-CCA and a trivial application of the reduction theorem gives the following property:

Proposition 6.1 *Let $\mathcal{AE}$ be an asymmetric encryption scheme. Then for any N, $\mathcal{AE}$ is $N - \delta_c$-IND-CCA iff $\mathcal{AE}$ is δ_c-IND-CCA.*

6.3 Soundness Result

The definition of Non EDY (NEDY) traces is exactly similar to the definition of NDY traces as the parsing algorithm α remains the same.

The main result is stated in the following theorem:

Theorem 6.1 *Let Π be a protocol. Let $\mathcal{AE}$ be the encryption scheme used in Π_c. If $\mathcal{AE}$ is δ_c-IND-CCA then for any concrete Adversary $\mathcal{A}_c$:*
$Pr\big[t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c\ ;\ t_c\ NEDY\big]$ *is negligible.*

The proof is close to the one of the previous soundness theorem, an adversary against the protocol can be transformed to an adversary against the encryption scheme as soon as the trace is NEDY.

- Parsing is done as before except that messages in S' cannot be decoded anymore. For that purpose, message in S' are stored as long as their formal equivalent to allow their parsing. Hence, it is possible to simulate the unfolding of the protocol.

- Messages that are NEDY are elements of the following grammar where m represents any message:

  $$n ::= N | \langle n, m\rangle | \langle m, n\rangle | \{n\}_{pk}$$

  If n is a nonce, winning the challenge can be done as before. If n is a pair, then one of its component is NEDY. Finally, if n is an encoding $\{n'\}_{pk}$ then the decryption oracle can be called to get n', otherwise $n \in S'$ but any message in S' is accessible in the trace and so is EDY.

Example 6.3 (Cipher Block Chaining) *Verification of protocols with CBC is possible when considering a bounded number of sessions (but the problem is in general NP-complete [9]). Thus, it is possible to verify a protocol with this hypothesis in the formal world, if the protocol is safe (regarding for example secret), then it is safe in the computational world.*

6.4 Extensions

Here, the weakening functions operate on messages/strings and produces a message/string. Although this hypothesis makes the results easier to understand, it is quite restrictive. This is why, we consider functions δ_c and δ_f which work on and produce sets of messages/strings. A similar soundness result can obtained in a straightforward way.

Example 6.4 (Extended CBC) *Let us consider an extension of CBC where the adversary is able to deduce a prefix as before but also a suffix. This is encoded in the following definition:*

$$\begin{aligned} \delta_f(\{\langle m, n\rangle\}_{pk}) &= \big\{\{m\}_{pk}, \{n\}_{pk}\big\} \\ \delta_c(\mathcal{E}(s.s', pk)) &= \big\{\mathcal{E}(s, pk), \mathcal{E}(s', pk)\big\} \end{aligned}$$

The EDY is modified with the two following inferences:

$$\frac{T \vdash \{\langle m, n\rangle\}_{pk}}{T \vdash \{m\}_{pk}} \qquad \frac{T \vdash \{\langle m, n\rangle\}_{pk}}{T \vdash \{n\}_{pk}}$$

Proof of the existence of a δ_c-IND-CCA encryption scheme for extended CBC is given in appendix E.

Conclusion

The main contributions of this paper are the following:

- A general definition of correctness of cryptographic primitives that generalizes known criteria such as IND-CPA and IND-CCA and allows to combine asymmetric, symmetric, and signature schemes as well as hash functions.

- A formal definition of a correctness criterion for hash functions.

- A theorem (Theorem 3.2) that allows to reduce the proof of correctness of a combination of cryptographic primitives to the correctness of each primitive individually.

- Based on this theorem, a proof of correctness of the Dolev-Yao model for protocols that may combine asymmetric, symmetric, and signature schemes as well as hash functions. The proof of our theorem makes some restrictions on the protocols that are in practice easily met. To our knowledge, this is the first time that correctness of Dolev-Yao is proved with such a combination of primitives.

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.

References

[1] 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)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[2] 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. ACM Press, 2003.

[3] 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, Lecture Notes in Comput. Sci., Vol. 1807*, pages 259–274. Springer, 2000.

[4] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Y. G. Desmedt, editor, *Proc. CRYPTO 94*, pages 341–358. Springer, 1994. Lecture Notes in Computer Science No. 839.

[5] 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. Springer-Verlag, 2000.

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

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

[8] 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.

[9] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

[10] J. A. Clark and J. L. Jacob. A survey of authentication protocol literature. Version 1.0, University of York, Department of Computer Science, Nov. 1997.

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

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

[13] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, Apr. 1984.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] Y. L. R. Janvier and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. Technical report, Verimag, Centre Équation, 38610 Gières, Oct. 2004.

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

[21] 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.

[22] `http://www-eva.imag.fr/`.

[23] 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.

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 long 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 4.2).

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 IND-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.

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

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,

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

Then, using lemma B.2, we get:

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

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 Theorem 5.1

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

$$p = Pr\big[t_c \leftarrow \mathcal{A}_c/\eta, \Pi_c\ ;\ t_c\ NDY\big]$$

We have the following majoration of p.

$$p \leq \big(2.Q(\eta) + 7\big).\mathbf{Adv}_{\mathcal{B}}^{\gamma}(\eta) + f(\eta) \qquad (1)$$

Where f is a negligible function. Using proposition 4.3, it is possible to deduce that the probability for $\mathcal{A}_c$ to produce a non Dolev-Yao trace is negligible.

Our Q-PASSH-CCA adversary $\mathcal{B}$ uses $\mathcal{A}_c$ as a subroutine and deduces a string solving its challenge (for example the challenge bit b or a new signature) as soon as $\mathcal{A}_c/\eta, \Pi_c$ produces a NDY trace. Using its own oracles, $\mathcal{B}$ simulates the protocol oracle Π_c (used by $\mathcal{A}_c$) and produces the pseudo formal trace in order to find a NDY message.

During its initialization, the adversary $\mathcal{B}$ randomly chooses an integer i between 0 and $Q(\eta)$. If $i \neq 0$, then the i^{th} nonce generated by $\mathcal{B}$ (denoted by N) is trapped. In order to answer queries from $\mathcal{A}_c$, $\mathcal{B}$ randomly generates identities for honest agents, nonces created by honest agents except N. $\mathcal{B}$ uses its challenge keys for their different keys. 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 $\mathcal{A}_c$ 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 $\mathcal{A}_c$ 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 its decryption oracles ($\mathcal{B}$ only has to decrypt messages encrypted using keys which inverse is known by an honest agent). 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, $\mathcal{A}_c$ stops. Then $\mathcal{A}_c$ 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, $\mathcal{B}$ also deduces b.
4. If m is a signature key, $\mathcal{B}$ can forge a new 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 $\mathcal{A}_c$ produces a NDY 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 $Q(\eta).Pr(E_1)$ and $Q(\eta).Pr(E_8)$. 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,

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

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

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

Hence, it is easy to obtain formula 1 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.

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

Thus, we have the following inequalities:

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

Proposition C.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.

D Existence of a CBC-IND-CCA encryption scheme

Let bs be a positive integer representing a block size. Let us consider an IND-CCA encryption scheme $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ and a SYM-CCA encryption scheme $\mathcal{AE}' = (\mathcal{KG}', \mathcal{E}', \mathcal{D}')$. Then, $\mathcal{AE}''$ is the encryption scheme $(\mathcal{KG}, \mathcal{E}'', \mathcal{D}'')$ where $\mathcal{E}''(s, pk)$ is defined as:

$$\mathcal{E}''(s, pk) = \mathcal{E}(s_1 \cdot k_1, pk) \cdot \mathcal{E}'(s_2 \cdot k_2, k_1) \cdot \ldots \cdot \mathcal{E}(s_n, k_{n-1})$$

when s is composed by blocks s_1 to s_n of size bs and k_i are freshly generated symmetric keys. The decoding algorithm allows to retrieve the original message by applying a similar modification.

This encryption scheme is not IND-CCA but is δ_c-IND-CCA. To prove that, let us assume that it is not the case. Then there exists an adversary $\mathcal{A}$ against this criterion. We use $\mathcal{A}$ to break Q-PAT-SYM-IND-CCA: for that purpose, $\mathcal{A}$ is executed and its call to oracles are simulated using IND-CCA oracles. Then, if $\mathcal{A}$ asks for the encryption of $s_1 \cdot \ldots \cdot s_n$, the new adversary asks for encryption of s_1, creation of a fresh challenge symmetric key SYM-CCA and so on. When $\mathcal{A}$ calls its decryption oracle, the argument m is not the prefix of any output of the new encryption oracle. Let us suppose that $m = m_1 \cdot ... \cdot m_n$. Then if m_1 has not been produced by the left-right asymmetric encryption oracle, it is possible to decrypt the whole message. Otherwise, let i be the minimal index such that m_i has not been produced by an encryption oracle. Then either m_i allows to win SYM-CCA/UNF or the m was not a correct encryption. Finally, if any m_i has been produced by an encryption oracle, then m is a prefix of the output of the new encryption oracle.

E Existence of an eCBC-IND-CCA encryption scheme

Let bs be a positive integer representing a block size. Let us consider an IND-CCA encryption scheme $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$. Then, $\mathcal{AE}'$ is the encryption scheme $(\mathcal{KG}, \mathcal{E}', \mathcal{D}')$ where $\mathcal{E}'(s, pk)$ is defined as:

$$\mathcal{E}'(s, pk) = \mathcal{E}(s_1, pk) \cdot \mathcal{E}(s_2, pk) \cdot ... \cdot \mathcal{E}(s_n, pk)$$

when s is composed by blocks s_1 to s_n of size bs. The decoding algorithm allows to retrieve the original message by applying a similar modification.

Then, $\mathcal{AE}'$ is not an IND-CCA encryption scheme but is a δ_c-IND-CCA encryption scheme. It is not IND-CCA as an adversary can submit message $0^{2.bs}$ and message $1^{2.bs}$ to the encryption oracle. The oracle outputs $\{b^{2.bs}\}_{pk} = \{b^{bs}\}_{pk}.\{b^{bs}\}_{pk}$ and so the adversary submits b^{bs} to the decryption oracles (this is possible as this message has not been produced by the encryption oracle) and obtains the value of bit b. The restriction made on the decryption oracle aims at forbidding such attacks.

If $\mathcal{AE}'$ is not a δ_c-IND-CCA encryption scheme, then there exists an adversary $\mathcal{A}$ against the criterion with δ_c. As usual, this adversary can be used to create an adversary against IND-CCA for algorithm $\mathcal{AE}$: oracles related to $\mathcal{A}$ are simulated using $\mathcal{AE}$ (this is trivial for encryption and possible for decryption because of the form of δ_c).