The Eighth Australasian Conference on Information Security and Privacy (ACISP '03).
(9 – 11 july 2003, Wollongong, Australia)
R. Safavi-Naini Ed. Springer-Verlag, LNCS 2727, pages 383–401.

Parallel Authentication and Public-Key Encryption

Josef Pieprzyk[1] and David Pointcheval[2]

[1] Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing, Macquarie University
Sydney, NSW 2109, AUSTRALIA
josef@ics.mq.edu.au

[2] École Normale Supérieure – Laboratoire d'informatique
45, rue d'Ulm, 75230 Paris Cedex 05, FRANCE
David.Pointcheval@ens.fr – www.di.ens.fr/users/pointche

Abstract. A parallel authentication and public-key encryption is introduced and exemplified on joint encryption and signing which compares favorably with sequential Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$) or Sign-then-Encrypt ($\mathcal{S}t\mathcal{E}$) schemes as far as both efficiency and security are concerned. A security model for signcryption, and thus joint encryption and signing, has been recently defined which considers possible attacks and security goals. Such a scheme is considered secure if the encryption part guarantees indistinguishability and the signature part prevents existential forgeries, for outsider but also insider adversaries. We propose two schemes of parallel signcryption, which are efficient alternative to Commit-then-Sign-and-Encrypt ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$). They are both provably secure in the random oracle model. The first one, called *generic parallel encrypt and sign*, is secure if the encryption scheme is semantically secure against chosen-ciphertext attacks and the signature scheme prevents existential forgeries against random-message attacks. The second scheme, called *optimal parallel encrypt and sign*, applies random oracles similar to the OAEP technique in order to achieve security using encryption and signature components with very weak security requirements — encryption is expected to be one-way under chosen-plaintext attacks while signature needs to be secure against universal forgeries under random-plaintext attack, that is actually the case for both the plain-RSA encryption and signature under the usual RSA assumption. Both proposals are generic in the sense that any suitable encryption and signature schemes (*i.e.* which simply achieve required security) can be used. Furthermore they allow both parallel encryption and signing, as well as parallel decryption and verification. Properties of parallel encrypt and sign schemes are considered and a new security standard for parallel signcryption is proposed.

Keywords: Signcryption, authentication, privacy, parallelization.

1 Introduction

The need for fast cryptographic transformations has never been so urgent as today when new multimedia applications such as distance learning, video on demand and TV channels delivery via Internet, interactive e-Commerce, etc. rely on secure transfer of large volumes of data. Typically data in transit needs to be cryptographically protected to provide either confidentiality and/or authenticity. As modern multimedia applications are run in real time, there are stringent requirements imposed on delay introduced by cryptography.

To speed up cryptographic transformations, we may apply two basic approaches. Firstly, we may design faster (symmetric or asymmetric) cryptographic algorithms. This option is not available most of the time. Once an algorithm becomes the standard or has been incorporated in hardware, users will be stuck with it for some time. Besides, the speed is typically determined by the number of rounds (in private key case) or by the size of the message (in public-key case). In the second approach, we can implement a parallel cryptographic system. Note that block ciphers (such as DES) have several operation modes from which some are sequential (like CBC and CFB) and some are parallel (like ECB and OFB).

The main idea is to take a large message block, divide it into blocks of the size determined by the applied cryptographic algorithm (in case of DES, the block size would be 64 bits) and apply the chaining using less expensive operations before the chained blocks are subject to cryptographic operation performed in parallel.

Consider what kind of cryptographic operations make sense for parallel execution. Encryption can be sped up by putting parallel encryption threads. For public-key cryptography, the chaining can be done by using hashing (hashing is much faster than public-key encryption). In the case of private-key cryptography, the chaining must be based on operations much faster than hashing (and encryption) such as bit-wise XOR.

The situation with digital signature looks differently. If the signer is a single person, then generation of parallel signatures (for the same message) is not very useful – one signature normally is enough. A plausible application of parallel signatures with a single signer is the case when the message is long lived and whose authenticity must be asserted even if one or more signature algorithms have been broken. More realistic application is when the same message is being signed by many co-signers in parallel as this is often required in group-oriented cryptography.

The most interesting case is, however, joint parallel encryption and signing. The scheme produces strings that can be seen from two different angles as designated verifier signatures or signed ciphertexts which can be verified by unique receiver. The parallel encryption and signing was introduced by [1]. Independently, the concept has been developed by the authors in this work. The both works can be seen as generalizations of the signcryption concept introduced by Zheng [17].

The work is structured as follows. Section 2 puts forward arguments for parallel encryption and signature, *a.k.a.* signcryption, and contrasts the approach with the previous ones. The security model for signcryption is presented in Section 3. The generic and optimal schemes for parallel signcryption are defined and analyzed in Sections 4 and 5, respectively. Section 6 concludes the work.

2 The Concept of Parallel Signing and Public-key Encryption

The encryption and signature algorithms are two basic cryptographic tools providing privacy and authenticity, respectively. However, in many applications it is desired to achieve both privacy and authenticity. Among many examples, we can mention transport of session keys between nodes of a network whose secrecy and authenticity is absolutely imperative to ensure secure data handling during sessions. Negotiations between two parties (businesses, institutions, countries, etc) typically have to be conducted in such a way that both the confidentiality and authenticity are guaranteed. The lack of confidentiality can be exploited by competitors. On the other hand, the lack of authenticity undermines the credibility of the negotiation process.

Both security goals are relatively easy to achieve if the cryptographic operations are performed using symmetric primitives. Note that in this setting, the fact that both parties share the same cryptographic key, means that everything not generated by one party had to be originated by the other one. In the public-key setting, cryptosystem can be applied either for privacy or authenticity. Clearly, when both goals have to be achieved, two cryptosystems have to be used in either Sign-then-Encrypt

($\mathcal{S}t\mathcal{E}$) or Encrypt-then-Sign ($\mathcal{E}t\mathcal{S}$) configuration [3]. Note that both configurations are inherently sequential.

Encryption and authentication are inseparable with conventional cryptography. The discovery of the public-key cryptography [7] divorced these two nicely coupled security goals and enabled party to choose either confidentiality (public-key encryption schemes) or authentication (signature schemes). Now in many applications, one would like to get both confidentiality and authentication but using public-key cryptography. Note that the come back to the conventional cryptography remains an unattractive option. Indeed, once the public-key cryptography is properly implemented with a secure and reliable public-key infrastructure (PKI), every single pair of parties may establish a secure communication (with confidentiality or/and authenticity) using the certificates of their public keys. Note also that within a single pair of parties, each party is uniquely identifiable by the public key of the receiver and the public key of the sender. Moreover, a signed ciphertext generated during communication between two parties can be explicitly attributed to a single sender whereas with symmetric cryptography this is not the case. Cryptograms are attributed implicitly — if I did not send this message, the other party did. However, it does not provide the important non-repudiation property. Furthermore, authentication fails when the secret key is shared by more than two parties.

Authenticated encryption has been studied by many authors mainly in the context of secret-key cryptography and message authentication code that is a symmetric-key equivalent to signature (see [3, 4]). Zheng [17] considered the problem in the context of public-key cryptography, with signcryption. The main problem considered in the paper [17] was how to design encryption and signature so that their concatenation maximizes savings of computing resources. A security model of parallel signcryption was defined recently in the work [1].

Our goal is to achieve the lower bound in terms of time necessary to perform authenticated encryption, and decryption as well, or

$$\begin{aligned}\text{time(parallel } \mathsf{encrypt} \ \& \ \mathsf{sign}) &\approx \max\{\text{time}(\mathsf{encrypt}),\text{time}(\mathsf{sign})\}\\ \text{and time(parallel } \mathsf{decrypt} \ \& \ \mathsf{verify}) &\approx \max\{\text{time}(\mathsf{decrypt}),\text{time}(\mathsf{verify})\}\end{aligned}$$

At best, one would expect that parallel encryption and sign will consume roughly the same time as the most time-consuming operation (either signing or encryption, for the joint encryption and signing, and either verifying or decrypting, for the joint decryption and verifying).

A hybrid approach called the envelop method can be used for authenticated encryption. Public key encryption and signature are used independently to generate cryptograms of both a secret key (which has been chosen at random by the sender) and a signature of a message. This method can concurrently encrypt and sign. The encryption is used for a secret key that is decrypted by the receiver. The secret key can be applied to encrypt a message using a symmetric encryption. This method can be simplified by independent encryption and signing (perhaps performed in parallel) of the same message. Note that in this case, weaknesses of encryption and signature schemes are likely to be preserved. In contrast, we show how to combine encryption and signature schemes so that they strengthen each other while they can still be run in parallel.

The Commit-then-Encrypt-and-Sign ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) can also be used to solve our problem [1], but it still requires strongly secure encryption and signature primitive to provide secure signcryption.

3 Model of Security

3.1 Signature Schemes

Description. A digital signature scheme SIGN consists of three algorithms [10]:

– GenSig, the *key generation algorithm* which, on input 1^k, where k is the security parameter, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys;
– Sig, the *signing algorithm* which receives a message m and the private key sk, and outputs a signature $\sigma = \mathsf{Sig}_{\mathsf{sk}}(m)$;
– Ver, the *verification algorithm* which receives a candidate signature σ, a message m, and a public key pk, and returns an answer $\mathsf{Ver}_{\mathsf{pk}}(m, \sigma)$ as to whether or not σ is a valid signature of m with respect to pk.

Security Notions. Attacks against signature schemes can be classified according to the goals of the adversary and to the resources that it can use. The goals are diverse and include:

– Disclosing the private key of the signer. This is the most drastic attack. It is termed the *total break*.
– Constructing an efficient algorithm that is able to sign any message with a significant probability of success. This is called the *universal forgery*. When the scheme prevents this kind of forgery it is said to be *Non Universally Forgeable* (NUF).
– Providing a single message/signature pair. This is called the *existential forgery*. When the scheme prevents this kind of forgery it is said to be *Non Existentially Forgeable* (NEF).

In terms of resources, we focus on two specific attacks against signature schemes: the *no-message attacks* and the *known-message attacks*. In the first scenario, the attacker only knows the public key pk of the signer. In the second, the attacker has access to a list of valid message/signature pairs. But this list may contain messages randomly and uniformly chosen, the attack is thus termed the *random-message attack* (RMA). Finally, the messages may be chosen, adaptively, by the adversary himself, we thus talk about the*chosen-message attack* (CMA).

In known-message attacks, one should point out that we consider a forgery of any valid signature that is not in the above list. This is the strongest security level, *a.k.a.* non-malleability [16].

3.2 Public-Key Encryption

Description. A public-key encryption scheme ENCRYPT is defined by three algorithms:

– GenEnc, the *key generation algorithm* which, on input 1^k, where k is the security parameter, produces a pair $(\mathsf{pk}, \mathsf{sk})$ of public and private keys;
– Enc, the *encryption algorithm* which, on input a plaintext m and a public key pk, outputs a ciphertext c;
– Dec, the *decryption algorithm* which, on input a ciphertext c and a private key sk, outputs the associated plaintext m (or $\perp$, if c is an invalid ciphertext).

Security Notions. The simplest security notion is *one-wayness* (OW): with public data only, an attacker cannot recover the whole plaintext m of a given ciphertext c. We denote by $\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{Encrypt}}(t)$ the maximum probability of success that an adversary can invert the encryption of a random plaintext in time t.

A stronger security notion has also been defined. It is the so-called *semantic security* (*a.k.a. indistinguishability of encryptions* [9], IND). If an attacker has some information about the plaintext, the view of the ciphertext should not leak any additional information. This security notion more formally considers the advantage an adversary can gain when trying to guess, between two messages, which one has been encrypted. In other words, an adversary is seen as a 2-stage Turing machine (A_1, A_2), and the advantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{Encrypt}}(\mathcal{A})$ should be negligible for any adversary, where

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{Encrypt}}(\mathcal{A}) = 2 \times \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k), (m_0, m_1, s) \leftarrow A_1(\mathsf{pk}), \\ b \in \{0,1\}, c = \mathsf{Enc}_{\mathsf{pk}}(m_b) : A_2(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

On the other hand, an attacker can use many kinds of attacks, depending on the information available to him. First, in the public-key setting, the adversary can encrypt any plaintext of his choice with the public key: this basic scenario is called the*chosen-plaintext attack*, and denoted by CPA. Extended scenarios allow the adversary a restricted or unrestricted access to various oracles. The main and strongest one is the decryption oracle which can be accessed adaptively in the *chosen-ciphertext* scenario, denoted CCA. There is the natural restriction that any query to this oracle should be different from the challenge ciphertext.

3.3 Joint Encryption and Signing: Signcryption

The following security model has been recently suggested and analyzed [1].

Description. A signcryption scheme SignCrypt is defined by three algorithms:

- Gen, the *key generation* algorithm which, for a security parameter k, outputs a pair of keys $(\mathsf{SDK}, \mathsf{VEK})$. SDK is the user's sign/decrypt key, which is kept secret, and VEK is the user's verify/encrypt key, which is made public.
- SigEnc, the *encryption and signing* algorithm which, for a message m, the public key of the receiver VEK_R and the private key of the sender SDK_S, produces a *signed–ciphertext* $c = \mathsf{SigEnc}_{\mathsf{SDK}_S, \mathsf{VEK}_R}(m)$.
- VerDec, the *decryption and verifying* algorithm which, for a *signed–ciphertext* c, the private key SDK_R of the receiver and the public key VEK_S of the sender, recovers the message $m = \mathsf{VerDec}_{\mathsf{VEK}_S, \mathsf{SDK}_R}(c)$. If this algorithm fails either to recover the message or to verify authenticity, it returns $\perp$.

Security Notions. For the security notions of a signcryption, one can combine the classical ones for signature [10] and encryption [2], under adaptive attacks. With an access to the public information, $\mathsf{PUB} = (\mathsf{VEK}_S, \mathsf{VEK}_R)$, and oracle access to the functionalities of both S and R (i.e. access to the *signcryption* and the *de-signcryption* oracles), the adversary should be able to break:

- authenticity (NEF) — come up with a valid *signed–ciphertext* of a new message, and thus provide an *existential forgery*;
- privacy (IND) — break the *indistinguishability* of *signed–ciphertexts*.

One should note that the adversary may be one of S or R themselves. But then, S may want to break the privacy, or R may want to break authenticity. If the signcryption scheme prevents existential forgeries and guarantees indistinguishability, in the above attack scenario, called *adaptive attacks* (AdA), we say the scheme is *secure*.

Definition 1. A signcryption scheme is **secure** if it achieves $\mathsf{IND}/\mathsf{NEF}$ under adaptive attacks.

Some Notations. Denote by $\mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A})$ the probability of success of an adversary in forging a new valid *signed–ciphertext*. Similarly, denote by $\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A})$ the advantage of an adversary in distinguishing *signed–ciphertexts*. Finally, denote $\mathsf{Win}^{\mathsf{secure}}_{\Pi}(\mathcal{A})$ as the maximum of these two values.

Let $\mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(t, q_1, q_2)$, $\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(t, q_1, q_2)$ and $\mathsf{Win}^{\mathsf{secure}}_{\mathsf{SignCrypt}}(t, q_1, q_2)$ be the respective probabilities for an adaptive adversary whose running time is bounded by t, while asking at most q_1 queries to the signcryption oracle and q_2 queries to the de-signcryption oracle.

4 Generic Parallel Signcryption

A trivial implementation of parallel signcryption could be as simple as encrypt and sign (with message recovery to allow parallel decryption and verification) the same message in parallel. This, of course, does not work as the signature reveals the message. Another classical solution could be the well-known *envelope technique* that first defines a secret session key. This key is encrypted under the public key encryption and is used, in parallel, to encrypt, under a symmetric encryption, the message and a signature on it. If one assumes that the symmetric encryption has a negligible cost (some may disagree with), then this allows parallel encryption and signing.The recipient first decrypts the session key, and then extracts the message and the signature. Only when all that operations have been completed, one can verify the signature. Therefore, decryption and verification cannot be done in parallel.

The Commit-then-Encrypt-and-Sign ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) [1] is a little bit better. Indeed, it first commits the message m, getting c the actual committed value, and d the decommitment. Then one encrypts d in e and signs c in s. The *signed–ciphertext* (e, c, s) can be de-signcrypted by first verifying (c, s) and decrypting e into d. The decommitment d finally helps to recover m. But the decommitment may not be as efficient as required.

Our idea exemplifies this technique, but with an efficient commitment scheme, in the random oracle model: given a message, we design a (2,2) Shamir secret sharing scheme [14] for which the secret is the message. Next, one of the shares is encrypted, the other is authenticated (in parallel). The perfectness of Shamir secret sharing guarantees that the knowledge of one of the shares provides no information (in the information-theoretical sense) about the secret.

4.1 Description

The building blocks are:

- an encryption scheme $\textsc{Encrypt} = (\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$,
- a signature scheme $\textsc{Sign} = (\mathsf{GenSig}, \mathsf{Sig}, \mathsf{Ver})$,
- a large k-bit prime p which defines the field $\mathbb{Z}_p$,

– a hash function $h : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ (assumed to behave like a random oracle [5]),
– k_1 and k_2, two security parameters such that $k = k_1 + k_2$.

Key Generation: $\mathsf{Gen}(1^k) = \mathsf{GenSig} \times \mathsf{GenEnc}(1^k)$
One first gets $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{GenSig}(1^k)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{GenEnc}(1^k)$. Then, one defines $\mathsf{SDK} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and $\mathsf{VEK} = (\mathsf{pk}_1, \mathsf{pk}_2)$.

Encrypt and Sign Algorithm: $\mathsf{SigEnc}_{\mathsf{SDK}_S, \mathsf{VEK}_R}(m)$

1. Let $m \in \{0,1\}^{k_1}$ be the message to be encrypted and signed. Choose a random integer $r \in \{0,1\}^{k_2}$ such that $(m\|r) \in \mathbb{Z}_p$ and compute $a = h(m\|r)$.
2. Form an instance of $(2,2)$ Shamir secret sharing scheme over $\mathbb{Z}_p$ with the polynomial $F(x) = (m\|r) + ax \bmod p$. Define two shares $s_1 = F(1)$ and $s_2 = F(2)$.
3. Calculate (in parallel) $c_1 = \mathsf{Enc}_{\mathsf{pk}_R}(s_1)$ and $c_2 = \mathsf{Sig}_{\mathsf{sk}_S}(s_2)$. The *signed–ciphertext* (c_1, c_2) is then dispatched to the receiver R.

Decrypt and Verify Algorithm: $\mathsf{VerDec}_{\mathsf{VEK}_S, \mathsf{SDK}_R}(c_1, c_2)$

1. Perform decryption and signature verification in parallel so, $t_1 = \mathsf{Dec}_{\mathsf{sk}_R}(c_1)$ and $t_2 = \mathsf{Ver}_{\mathsf{pk}_S}(c_2)$. Note that both the decryption Dec and verification Ver algorithms return integers in $\mathbb{Z}_p$, unless some failure occurs. Indeed, it is possible that Dec returns $\perp$ if it decides that the cryptogram is invalid. Similarly, Ver returns a message (signature with message recovery), or $\perp$ if the signature is invalid. In case of one failure, the decryption and verifying algorithm VerDec returns $\perp$ and stops.
2. Knowing two points $(1, t_1)$ and $(2, t_2)$, use the Lagrange interpolation and find the polynomial $\tilde{F}(x) = a_0 + a_1 x \bmod p$.
3. Check whether $a_1 = h(a_0)$. If the check holds, the algorithm extracts m from a_0 (note that $a_0 = (m\|r)$) and returns m. Otherwise, the algorithm outputs $\perp$.

4.2 Security of Generic Scheme

Theorem 2. *If the encryption scheme is* IND-CCA *and the signature scheme is* NEF-RMA*, then the generic parallel signcryption scheme is secure (*IND/NEF-AdA*).*

More precisely, one can claim the following result:

Lemma 3. *Let us consider an* AdA *adversary $\mathcal{A}$ against* IND *and* NEF *of the generic parallel signcryption, with a running time bounded by t, while asking q_h queries to the random oracle h, and q_1 and q_2 queries to the signcryption and de-signcryption oracles respectively. Then, the winning probability of this adversary is bounded by*

$$2 \times \mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{Encrypt}}(t', q_2) + 6 \times \mathsf{Succ}^{\mathsf{nef-rma}}_{\mathsf{Sign}}(t', q_1) + (5 + 2q_1) \times \frac{q_h + q_1 + q_2}{2^{k_2}},$$

with $t' \leq t + (q_1 + q_2)(\tau + \mathcal{O}(1))$, where τ denotes the maximal running time of the encryption, decryption, signing and verification algorithms.

The proof can be found in the Appendix A.

4.3 Properties

From the efficiency point of view, this generic scheme is almost optimal since only one hash value and two additions are required before the parallel encryption and signature processes. The reverse process reaches the same kind of optimality.

However, the security requirements of the basic schemes, the encryption scheme Encrypt and the signature scheme Sign, are very strong. Indeed, the encryption scheme is required to be semantically secure against chosen-ciphertext attack and the signature scheme must already prevent existential forgeries.

5 Optimal Parallel Signcryption

Adding a kind of OAEP technique [6], we can improve the generic scheme, in the sense that we can weaken the security requirements of the basic primitives. The new proposal just requires the encryption scheme to be deterministic and one-way against chosen-plaintext attack, which is a very weak security requirement (even the plain-RSA [12] achieves it under the RSA assumption). The signature scheme is required to prevent universal forgeries under random-message attack (the plain-RSA signature also achieves this security level).

5.1 Description

The building blocks are:

- an encryption scheme Encrypt $= (\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$,
- a signature scheme Sign $= (\mathsf{GenSig}, \mathsf{Sig}, \mathsf{Ver})$,
- a large k-bit prime p which defines the field $\mathbb{Z}_p$,
- k_1 and k_2, two security parameters such that $k = k_1 + k_2$.
- hash functions (assumed to behave like random oracles [5]),

$$f : \{0,1\}^k \to \{0,1\}^k, g : \{0,1\}^k \to \{0,1\}^k \text{ and } h : \{0,1\}^{k+k_1} \to \{0,1\}^{k_2}.$$

Key Generation: $\mathsf{Gen}(1^k) = \mathsf{GenSig} \times \mathsf{GenEnc}(1^k)$
One first gets $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{GenSig}(1^k)$ and $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{GenEnc}(1^k)$. Then, one defines $\mathsf{SDK} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and $\mathsf{VEK} = (\mathsf{pk}_1, \mathsf{pk}_2)$.

Encrypt and Sign Algorithm: $\mathsf{SigEnc}_{\mathsf{SDK}_S, \mathsf{VEK}_R}(m)$

1. Let $m \in \mathbb{Z}_p$ be the message to be encrypted and signed. Choose a random integer $r \in \{0,1\}^{k_1}$ and compute $a = h(m\|r)$.
2. Form an instance of a $(2,2)$ Shamir secret sharing scheme over $\mathbb{Z}_p$ with the polynomial $F(x) = (a\|r) + mx \bmod p$. Define two shares $s_1 = F(1)$ and $s_2 = F(2)$.
3. Compute the transform $r_1 = s_1 \oplus f(s_2)$ and $r_2 = s_2 \oplus g(r_1)$.
4. Calculate (in parallel) $c_1 = \mathsf{Enc}_{\mathsf{pk}_R}(r_1)$ and $c_2 = \mathsf{Sig}_{\mathsf{sk}_S}(r_2)$. The *signed–ciphertext* (c_1, c_2) is then dispatched to the receiver R.

Decrypt and Verify Algorithm: $\mathsf{VerDec}_{\mathsf{VEK}_S,\mathsf{SDK}_R}(c_1, c_2)$

1. Perform decryption and signature verification in parallel so, $u_1 = \mathsf{Dec}_{\mathsf{sk}_R}(c_1)$ and $u_2 = \mathsf{Ver}_{\mathsf{pk}_S}(c_2)$. Note that both the decryption Dec and verification Ver algorithms return integers in $\mathbb{Z}_p$, unless some failure occurs. Indeed, it is possible that Dec returns $\perp$ if it decides that the cryptogram is invalid. Similarly, Ver returns a message (signature with message recovery), or $\perp$ if the signature is invalid. In case of one failure, the decryption and verifying algorithm VerDec returns $\perp$ and stops.
2. Compute the inversion $t_2 = u_2 \oplus g(u_1)$ and $t_1 = u_1 \oplus f(t_2)$.
3. Knowing two points $(1, t_1)$ and $(2, t_2)$, use the Lagrange interpolation and find the polynomial $\tilde{F}(x) = a_0 + a_1 x \bmod p$.
4. Extract r from a_0 and check whether $h(a_1 \| r) \| r = a_0$. If the check holds, the algorithm returns a_1, to be m. Otherwise, the algorithm outputs $\perp$.

5.2 Security Analysis

Theorem 4. *If the encryption scheme is deterministic and* OW-CPA*, and the signature scheme is* NUF-RMA*, then the optimal parallel signcryption scheme is secure (*IND/NEF-AdA*).*

About this theorem, one can claim a more precise result:

Lemma 5. *Let us consider an* AdA *adversary $\mathcal{A}$ against* IND *and* NEF *of the optimal parallel signcryption scheme, with a running time bounded by t, while asking q queries to the random oracles, and q_1 and q_2 queries to the signcryption and de-signcryption oracles, respectively. Then the winning probability of this adversary is bounded by*

$$\mathsf{Succ}^{\mathsf{ow-cpa}}_{\mathsf{Encrypt}}(t) + Q \times \mathsf{Succ}^{\mathsf{nuf-rma}}_{\mathsf{Sign}}(t', Q) + \frac{1}{2^{k_2}} \times (1 + 4Q^2 + 3q_2 + q_1) + \frac{q}{2^{k_1}},$$

with $t' \leq t + Q(\tau + \mathcal{O}(1))$, where τ denotes the maximal running time of the encryption, decryption, signing and verification algorithms, and $Q = q + q_1 + q_2$.

Proof. The proof is similar to the proof of the Lemma 3. It is therefore also divided into two parts. In the first one, we are going to show that the scheme meets IND-AdA, but under the assumption that it meets NEF-AdA. The second part deals with NEF, and shows that it actually meets NEF-AdA.

In the proof, when one calls to f, g, or h, if the query has already been asked, or the answer has already been defined by the simulation, the same answer is returned, otherwise a random value in the according range is given. Of course, one has to be careful when one defines an answer of a random oracle:

– this answer must not have already been defined
– the answer must be uniformly distributed

Furthermore, we denote by q_F, q_G and q_H the number of answers defined for f, g and h, respectively. We will see at the end of the simulation the relations with q_f, q_g and q_h, the number of queries directly asked by the adversary (thus $q = q_f + q_g + q_h$).

Indistinguishability: IND. Let us assume that after q_1 queries to oracle SigEnc and q_2 queries to oracle VerDec, after having chosen a pair of message m_0 and m_1, and received a *signed–ciphertext* (c_1, c_2) of either m_0 or m_1, say m_b, an adversary $\mathcal{A}$ outputs a bit d which is equal to b with advantage ε: $\Pr[d = b] = (1 + \varepsilon)/2$.

Let us first remark that because of the randomness of the random oracles f and g, to get any information about the bit b (and thus about the encrypted and signed message), an adversary must have got some information about s_1 or s_2 from either the *signed–ciphertext*, or from the plaintext and the random tape.

The former case is only possible if the adversary asks for r_1 to the oracle g (otherwise it has no information about s_2, and s_1 neither). This event is denoted AskG. The latter case means that the adversary asked either $h(m_0\|r)$ or $h(m_1\|r)$. This event is denoted AskR. Consequently,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) &= 2\Pr[d = b] - 1 \\
&= 2\Pr[d = b \wedge (\mathsf{AskG} \vee \mathsf{AskR})] + 2\Pr[d = b \wedge \neg(\mathsf{AskG} \vee \neg\mathsf{AskR})] - 1 \\
&\leq 2\Pr[\mathsf{AskG} \vee \mathsf{AskR}] + \Pr[\neg(\mathsf{AskG} \vee \mathsf{AskR})] - 1 = \Pr[\mathsf{AskG} \vee \mathsf{AskR}] \\
&\leq \Pr[\mathsf{AskG}] + \Pr[\mathsf{AskR} \mid \neg\mathsf{AskG}] \leq \Pr[\mathsf{AskG}] + \frac{q_h}{2^{k_1}}
\end{aligned}$$

Therefore, r_1 the plaintext of c_1 necessarily appears in the queries asked to g. For each query asked to g, one runs the deterministic encryption algorithm and therefore can find the plaintext of a given c_1: we may use this adversary $\mathcal{A}$ to break $\mathsf{OW\text{-}CPA}$ of the encryption scheme $(\mathsf{GenEnc}, \mathsf{Enc}, \mathsf{Dec})$. To achieve this aim, we design a simulator $\mathcal{B}$ which is given the private/public keys $(\mathsf{sk}_S, \mathsf{pk}_S)$ for the signature scheme, but is just further given the public key pk_R of the encryption scheme.

- $\mathcal{B}$ is given a ciphertext c (of a random message) to decrypt under the encryption scheme Encrypt, and then runs $\mathcal{A}$.
- When $\mathcal{B}$ receives the pair of messages m_0 and m_1 from $\mathcal{A}$, it defines c (the challenge ciphertext to decrypt) as c_1, and randomly chooses $r_2 = t_2$, that it can sign using the private key of the signature scheme, to produce c_2. It therefore sends the pair (c_1, c_2) as a *signed–ciphertext* of m_b (for some bit b). Finally, the adversary $\mathcal{A}$ follows in its attack.
- Any call by $\mathcal{A}$ to the oracle SigEnc can be simply answered by $\mathcal{B}$ using the private key of the signature scheme, and the public key of the encryption scheme. It makes one more call to each of the random oracles f, g and h.
- Before simulating the oracle VerDec, let us explain how one deals with h-queries. Indeed, a list Λ_h is managed. For any query $h(m\|r)$, one anticipates the signcryption:

  $$H = h(m\|r) \quad a_0 = H\|r \quad t_1 = a_0 + m \bmod p \quad t_2 = a_0 + 2m \bmod p.$$

  Then, $u_1 = t_1 \oplus f(t_2)$ and $u_2 = t_2 \oplus g(u_1)$ (using the simulations of f and g). Eventually, one stores $(m, r, H, u_1, u_2, t_1, t_2)$ in Λ_h.
- Any call by $\mathcal{A}$ to the oracle VerDec can be simulated using the queries-answers of the random oracles. Indeed, to a query (c_1', c_2'), one first gets u_2' from c_2', thanks to the public key of the signature scheme ($u_2' = \mathsf{Ver}_{\mathsf{pk}_R}(c_2')$). Then, one looks up into Λ_h for tuples $(m, r, H, u_1, u_2', t_1, t_2)$. Then, one checks whether one of the u_1 is really encrypted in c_1', thanks to the deterministic property of the encryption. If no tuple is found, the simulator outputs $\perp$, considering it is a wrong *signed–ciphertext*. Otherwise, the simulator returns m, to be the plaintext.

For all the *signed–ciphertexts* correctly constructed (with $s'_2 = t'_2$ asked to f, $r'_1 = u'_1$ asked to g and $(m'\|r')$ asked to h), the simulation gets back the message. But the adversary may produce a valid *signed–ciphertext* without asking $h(m'\|r')$ required by the above simulation. In that sole case, the simulation may not be perfect.

First, let us assume that $(m'\|r')$ has not been asked to h. Then, either $(m'\|r') \neq (m\|r)$ (the pair involved in the challenge *signed–ciphertext*) then H is totally random. The probability for $H\|r'$ to match with a'_0 is less than 2^{-k_2}. Or $(m'\|r') = (m\|r)$. Since all the process to produce r'_1 and r'_2 is deterministic, $r'_1 = r_1$ and $r'_2 = r_2$, the same as in the challenge *signed–ciphertext*. The encryption is deterministic, then $c'_1 = c_1$. Therefore, either the adversary produced a new *signed–ciphertext*, which is bounded by $\mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A})$, or the simulation of the sign-crypt oracle signed twice the value t_2 involved in the challenge *signed–ciphertext*, which is upper-bounded by $q_1/2^{k_2}$, because of the randomness of r Therefore, the probability that some $(m'\|r')$ is equal to $(m\|r)$ is upper-bounded by $\mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) + q_1/2^{k_2}$. As a consequence, $(m'\|r')$ has been likely asked to h, otherwise the simulation just fails with probability less than 2^{-k_2}.

Then, the simulator can extract t'_1 and t'_2. But because of the randomness of H, the probability for t'_2 to be equal to t_2 (the one involved in the challenge *signed–ciphertext*) is less than $q_H/2^{-k_2}$. If it is not the case, and t'_2 not asked to f, then the probability for the resulting u'_1 to be in the list of the queries asked to g (explicitly or implicitly) is less than $(q_G + 1)/2^k$. If it is not the case, the probability for u'_2 to match with u_2 (the one involved in the challenge *signed–ciphertext*) is less than 2^{-k}.

Therefore, the probability that the simulation is not correctly decrypted (provided that no signature forgery occurs and there is no double signatures on r_2) is less than

$$2^{-k_2} + q_H \cdot 2^{-k_2} + (q_G + 1)\cdot 2^{-k} + 2^{-k} = \frac{q_H + 1}{2^{k_2}} + \frac{q_G + 2}{2^k} \leq \frac{q_G + q_H + 3}{2^{k_2}}.$$

And thus, the simulations are all perfect with the probability greater than

$$1 - \frac{q_2 \cdot (q_G + q_H + 3) + q_1}{2^{k_2}} - \mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}).$$

If all the decryption simulations are correct (no occurrence of the event BadD), we have seen that with a good probability the plaintext c_1, and thus of c, appears in the queries asked to g, which is immediately detected thanks to the deterministic property of the encryption scheme so

$$\begin{aligned}&\Pr[\mathsf{AskG} \mid \neg\mathsf{BadD}] \geq \Pr[\mathsf{AskG}] - \Pr[\mathsf{BadD}]\\ &\geq \mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) - \frac{q_h}{2^{k_1}} - \frac{q_2 \cdot (q_G + q_H + 3) + q_1}{2^{k_2}} - \mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}).\end{aligned}$$

The expression upper-bounds the advantage of $\mathcal{A}$ in $\mathsf{IND\text{-}AdA}$ by

$$\mathsf{Succ}^{\mathsf{ow-cpa}}_{\mathsf{Encrypt}}(t') + \frac{q_h}{2^{k_1}} + \frac{q_2 \cdot (q_G + q_H + 3) + q_1}{2^{k_2}} + \mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}),$$

where $q_F \leq q_f + q_H$, $q_G \leq q_g + q_H$, and $q_H \leq q_h + q_1$.

Non Existential Forgery: NEF. Let us assume that after q_1 queries to oracle SigEnc and q_2 queries to oracle VerDec, an adversary $\mathcal{A}$ outputs (or asks to VerDec) a new *signed–ciphertext* (c_1, c_2) which is valid with probability ε. We will use this adversary to perform a universal forgery, as thus produces a new signature on a designated random message (under a known random-message attack) against the signature scheme Sign.

To achieve this aim, we design a simulator $\mathcal{B}$ which has access to a list of message-signature pairs, produced by the signing oracle (the messages are assumed to have been randomly drawn in $\mathbb{Z}_p$, but not chosen by the adversary). Note that a valid *signed–ciphertext* must satisfy the equality $h(m\|r)\|r = a_0$. Therefore, the probability to output such a valid *signed–ciphertext* without asking $h(m\|r)$ is smaller than 2^{-k_2}: with probability greater than $\varepsilon - 2^{-k_2}$, this query $(m\|r)$ has been asked to h.

The simulator $\mathcal{B}$ is given the private/public keys $(\mathsf{sk}_R, \mathsf{pk}_R)$ for the encryption scheme, but is just given the public key pk_S of the signature scheme. It is furthermore given a list of q_H message-signature (M, S), in which we assume that the message on which one has to produce a new signature is randomly located, say M_i.

– For any new query $(m\|r)$ asked to h (by the adversary, or by our simulations of SigEnc and VerDec), a new valid message-signature pair (M, S) is taken from the list. Then, one chooses a random ρ, defines $h(m\|r) \leftarrow \rho$ and sets

$$s_1 \leftarrow \rho\|r + m \bmod p \quad s_2 \leftarrow \rho\|r + 2m \bmod p \quad r_1 \leftarrow s_1 \oplus f(s_2).$$

One eventually defines $g(r_1) \leftarrow s_2 \oplus M$, which is a random value, since M is randomly distributed. It may fail if $g(r_1)$ has already been defined. But because of the random choice of ρ this may just occurs with probability less than $q_G/2^{k_2}$. Remark that the comments above would be wrong if the value $h(m\|r)$ would not be a new random value, but a value already defined by the simulation. But one can remark that no answer for h is defined by a simulator in the proof, but all using this simulation.

– Any query m by $\mathcal{A}$ to the oracle SigEnc can be simulated, thanks to above simulation of h. Indeed, one simply chooses a random r, asks for $h(m\|r)$. Then the signature S involved in the pair (M, S) used for the h simulation is a signature c_2 of $r_2 = M$. Using the public key pk_R of the encryption scheme, one can encrypt r_1 to obtain c_1. The pair (c_1, c_2) is a valid *signed–ciphertext* of m.

– Any call by $\mathcal{A}$ to the oracle VerDec can be simulated using the private key sk_R of the encryption scheme and the public key pk_S of the signature scheme.

Finally, the adversary $\mathcal{A}$ produces a new *signed–ciphertext* (c_1, c_2) which is valid with probability greater than ε, unless the above simulation of h failed (such a failure happens with probability upper bounded by $q_H q_G/2^{k_2}$).

Furthermore, this *signed–ciphertext* is involved in one of the h-queries, but with probability $1/2^{k_2}$. With probability $1/q_H$, this *signed–ciphertext* is involved in the i-th query to the h-oracle: c_2 is a valid signature of M_i. But either this is a new signature, or it was already involved in a *signed–ciphertext* (c_1', c_2') produced by SigEnc. But in this latter case, since $c_2 = c_2'$, necessarily $c_1 \neq c_1'$. But because of the determinism of the encryption scheme, it means that $u_1 \neq u_1'$, and then the redundancy may hold but with probability less than $q_G q_H/2^{k_2}$.

Finally, the probability for $\mathcal{B}$ to produce a new valid signature of M_i is greater than

$$\frac{1}{q_H} \times \left(\varepsilon - \frac{2q_G q_H + 1}{2^{k_2}} \right).$$

Furthermore, one can easily see that $q_F = q_f + q_H$ and $q_G = q_g + q_H$, where $q_H \leq q_h + q_1 + q_2$. $\square$

6 Conclusion

We have introduced parallel signcryption schemes which are superior to well-studied sequential Sign-then-Encrypt or Encrypt-then-Sign schemes, or any other combination, in term of their efficiency, since they allow parallel signature and encryption as well as parallel decryption and verification.

The optimal scheme is especially attractive as it is secure using a weak encryption (i.e., one-way under chosen-plaintext attack) and a weak signature scheme (i.e., signature is required to be secure against universal forgeries under random-message attack).

It has been shown that the OAEP technique which was applied for encryption [6] (with other recent studies [15, 8]) can also be used for parallel signcryption. The OAEP technique was incorporated into RSA Security standards for encryption (for details see the description of PKCS#1 v2.1 [13].) Parallel signcryption is potentially a candidate for the third missing standard for parallel authenticated encryption.

The message redundancy of the parallel signcryption scheme can be measured as the ratio of the *signed–ciphertext* length to the message length. It is easy to see that both schemes considered expand the length of *signed–ciphertext* by the factor of more than 2. It is an interesting question whether the redundancy can be reduced while leaving security conclusions intact.

Acknowledgement

The work was partially supported by Australia Research Council grants A00103078 and DP0345366.

References

1. J. H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signatures and Encryption. In *Eurocrypt '02*, LNCS 2332, pages 83–107. Springer-Verlag, Berlin, 2002.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
3. M. Bellare and C. Namprempre. Authenticated Encryption: Notions and Constructions. In *Asiacrypt '00*, LNCS 1976. Springer-Verlag, Berlin, 2000.
4. M. Bellare and P. Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancies in Plaintexts for Efficient Cryptography. In *Asiacrypt '00*, LNCS 1976. Springer-Verlag, Berlin, 2000.
5. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
6. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
7. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
8. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '01*, LNCS 2139, pages 260–274. Springer-Verlag, Berlin, 2001.
9. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
10. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
11. J. Pieprzyk and D. Pointcheval. Parallel Authentication and Public-Key Encryption. In *ACISP '03*, LNCS. Springer-Verlag, Berlin, 2003.
Full version available from `http://www.di.ens.fr/users/pointche/`.
12. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

13. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.
Available from `http://www.rsa.com/rsalabs/pubs/PKCS/`.
14. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, November 1979.
15. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.
16. J. Malone-Lee, D. Pointcheval, N. Smart, and J. Stern. Flaws in Applying Proof Methodologies to Signature Schemes. In *Crypto '02*, LNCS 2442, pages 93–110. Springer-Verlag, Berlin, 2002.
17. Y. Zheng. Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In *Crypto '97*, LNCS 1294. Springer-Verlag, Berlin, 1997.

A Proof of Lemma 3

The proof is divided into two parts. In the first one, we are going to show that the scheme meets IND-AdA. The second part deals with NEF.

In all the proof, when one calls to h, if the query has already been asked, or the answer has already been defined by the simulation, the same answer is returned, otherwise a random value in $\mathbb{Z}_p$ is given. Of course, one has to be careful when one defines an answer of a random oracle:

– this answer must not have already been defined
– the answer must be uniformly distributed

Furthermore, we denote by q_H the number of answers defined for h. We will see at the end of the simulation the relation with q_h.

A.1 Indistinguishability: IND

Let us assume that after q_1 queries to oracle SigEnc and $q = q_2$ queries to oracle VerDec, after having chosen a pair of message m_0 and m_1, and received a *signed–ciphertext* (c_1, c_2) of either m_0 or m_1, say m_b, an adversary $\mathcal{A}$ outputs a bit d which is equal to b with advantage ε: $\Pr[d = b] = (1 + \varepsilon)/2$.

We furthermore assume that all the valid signatures involved in the *signed–ciphertexts* have been produced by our simulators, but maybe with probability less than ν. This case of signature forgeries, which event is denoted by Forge, will be studied later.

$$\nu = \Pr[\mathsf{Forge}].$$

We will use this adversary to break the IND-CCA security of the encryption scheme Encrypt. To achieve this aim, we design a simulator $\mathcal{B}$ which has access to the decryption oracle Dec. It is given the private/public keys $(\mathsf{sk}_S, \mathsf{pk}_S)$ for the signature scheme, but is just further given the public key pk_R of the encryption scheme.

– Any call by $\mathcal{A}$ to the oracle SigEnc can be simply answered by $\mathcal{B}$ using the private key of the signature scheme, and the public key of the encryption scheme.
– Any call by $\mathcal{A}$ to the oracle VerDec can be simulated using the decryption oracle Dec access. Indeed, to a query (d_1, d_2), one first asks the query d_1 to the oracle Dec and obtains t_1. Thanks to the public key of the signature scheme, one can get t_2 from d_2. This is enough to check the validity of the *signed–ciphertext* (d_1, d_2) and to decrypt it. Unless d_1 is the same as the challenge ciphertext c_1. In this case, necessarily $d_2 \neq c_2$, and the *signed–ciphertext* is rejected. We will denote by BadD the event that such a *signed–ciphertext* is wrongly rejected. Under the assumption $\neg$Forge, we hope BadD to be negligible.

When $\mathcal{B}$ receives the pair of messages m_0 and m_1 from $\mathcal{A}$, it randomly chooses two random integers r_0 and r_1 to produce two new messages for the encryption scheme:

$$\begin{aligned} M_0 &\leftarrow (m_0\|r_0) + h(m_0\|r_0) \bmod p \\ M_1 &\leftarrow (m_1\|r_1) + h(m_1\|r_1) \bmod p \end{aligned}$$

$\mathcal{B}$ receives the ciphertext c_1 of M_b, and has to guess the bit b, with the help of $\mathcal{A}$. For that, it chooses a random bit b' (hoping it to be equal to b) and defines

$$s_2 \leftarrow (m_{b'}\|r_{b'}) + 2h(m_{b'}\|r_{b'}) \bmod p.$$

Then, it can sign it using the private key of the signature scheme, to produce c_2. It therefore sends the pair (c_1, c_2) as a *signed–ciphertext* of m_b (for the unknown bit b). Finally, the adversary $\mathcal{A}$ ends its attack, returning a bit d, and $\mathcal{B}$ forwards it as its final answer.

Now, we study the advantage of the adversary $\mathcal{B}$ in breaking $\mathsf{IND\text{-}CCA}$ of the encryption scheme, which is

$$\begin{aligned} &\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{Encrypt}}(\mathcal{B}) = 2\Pr[d=b] - 1 \geq 2\Pr[d=b \,|\, \neg\mathsf{Forge}] - 2\Pr[\mathsf{Forge}] - 1 \\ &\geq 2\Pr[d=b \wedge \neg\mathsf{BadD} \,|\, \neg\mathsf{Forge}] - 1 - 2\nu \\ &\geq 2\Pr[d=b \,|\, \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] - 2\Pr[\mathsf{BadD} \,|\, \neg\mathsf{Forge}] - 1 - 2\nu \\ &\geq \Pr[d=b \,|\, b=b' \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] + \Pr[d=b \,|\, b \neq b' \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] \\ &-2\Pr[\mathsf{BadD} \,|\, \neg\mathsf{Forge}] - 1 - 2\nu \end{aligned}$$

Let us now study each term. One should first note that

$$\begin{aligned} &\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) = 2\Pr[d=b \,|\, b'=b] - 1 \\ &\leq 2\Pr[d=b \,|\, \neg\mathsf{Forge} \wedge b'=b] + 2\Pr[\mathsf{Forge} \,|\, b'=b] - 1 \\ &\leq 2\Pr[d=b \,|\, b'=b \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] + 2\Pr[\mathsf{BadD} \,|\, \neg\mathsf{Forge}] + 2\nu - 1 \end{aligned}$$

Let us now focus on the second term in the inequality, by defining AskH to be the event that the adversary $\mathcal{A}$ either asks $(m_0\|r_0)$ or $(m_1\|r_1)$ to the random oracle h. It is equal to

$$\begin{aligned} &\Pr[d=b \,|\, b' \neq b \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] \geq \\ &\Pr[d=b \,|\, b' \neq b \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge} \wedge \neg\mathsf{AskH}] \\ &\times \mathsf{pr}[\neg\mathsf{AskH} | b' \neq b \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge}] \end{aligned}$$

Clearly, in the case that $b' \neq b$, the adversary may just have some information (in the theoretical sense) about

$$\begin{aligned} M_b &= (m_b\|r_b) + h(m_b\|r_b) \bmod p \\ s_2 &= (m_{b'}\|r_{b'}) + 2h(m_{b'}\|r_{b'}) \bmod p. \end{aligned}$$

But without the event AskH, the hash values perfectly hide the first part, and therefore the answer of $\mathcal{A}$ is independent of b (a random variable):

$$\Pr[d=b \,|\, b' \neq b \wedge \neg\mathsf{BadD} \wedge \neg\mathsf{Forge} \wedge \neg\mathsf{AskH}] = \frac{1}{2}.$$

On the other hand, as said above, the $h(m_i\|r_i)$ perfectly hide the $(m_i\|r_i)$, for $i = 0, 1$, and therefore one cannot get any information about the random values r_0 and r_1 without luck: AskH with probability less than $2q_H/2^{k_2}$.

Finally, let us study the probability of a wrong decryption. For that, we focus on executions in which the adversary does not perform any signature forgery. Therefore, d_2 is the signature of t_2, a uniformly distributed element, independently from the wishes of the adversary. A bad decryption may just occur if $d_1 = c_1$, and thus is the encryption of M_b, uniformly distributed as well, independently from the wishes of the adversary. The probability that a t_2 (signed by our simulation) provides the required redundancy, $t_2 - M_b = h(2M_b - t_2)$, is less than $q_1/2^k$. Hence,

$$\Pr[\mathsf{BadD} \mid \neg\mathsf{Forge}] \leq \frac{q_1}{2^k}.$$

This concludes the proofs:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{Encrypt}}(\mathcal{B}) &\geq \frac{1}{2} \cdot \left(\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) - 2\Pr[\mathsf{BadD} \mid \neg\mathsf{Forge}] - 2\nu + 1\right) \\
&\quad + \frac{1}{2} \cdot \left(1 - \frac{2q_H}{2^{k_2}}\right) - 2\Pr[\mathsf{BadD} \mid \neg\mathsf{Forge}] - 1 - 2\nu \\
&\geq \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) - 3\Pr[\mathsf{BadD} \mid \neg\mathsf{Forge}] - 3\nu - \frac{q_H}{2^{k_2}} \\
&\geq \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) - \frac{3q_1}{2^k} - 3\nu - \frac{q_H}{2^{k_2}}
\end{aligned}$$

where $q_H \leq q_h + q_1 + q_2$:

$$\mathsf{Adv}^{\mathsf{ind-ada}}_{\mathsf{SignCrypt}}(t, q_1, q_2) \leq 2 \times \mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{Encrypt}}(t', q_2) + \frac{6q_1}{2^k} + \frac{2(q_h + q_1 + q_2)}{2^{k_2}} + 6\nu,$$

where t' is the announced bound.

A.2 Non Existential Forgery: NEF

Let us assume that after $q = q_1$ queries to oracle SigEnc and q_2 queries to oracle VerDec, an adversary $\mathcal{A}$ outputs (or asks to the oracle VerDec) a new *signed–ciphertext* (c_1, c_2) which is valid with probability ν. We use this adversary to perform an existential forgery (under a random-message attack) against the signature scheme Sign.

To achieve this aim, we design a simulator $\mathcal{B}$ which has access to a list of message-signature pairs, produced by the signing oracle (the messages are assumed to have been randomly drawn in $\mathbb{Z}_p$, but not chosen by the adversary). It is given the private/public keys $(\mathsf{pk}_R, \mathsf{sk}_R)$ for the encryption scheme, but is just further given the public key pk_S of the signature scheme.

– Any query m by $\mathcal{A}$ to the oracle SigEnc can be simulated using a new valid message-signature pair (M, S), for the signature scheme. Indeed, M is defined to be s_2 and S is defined to be c_2. Then, one chooses a random r. Since

$$s_2 = (m\|r) + 2h(m\|r) \bmod p = M,$$

one has defined the random oracle at the point $(m\|r)$ (unless it has already been done, which then raises event BadH):

$$h(m\|r) \leftarrow \frac{M - (m\|r)}{2} \bmod p$$

and therefore,

$$s_1 \leftarrow (m\|r) + h(m\|r) = \frac{M + (m\|r)}{2} \bmod p.$$

Using the public key of the encryption scheme, one can encrypt s_1 to obtain c_1. The pair (c_1, c_2) is a valid *signed–ciphertext* of m.

– Any call by $\mathcal{A}$ to the oracle VerDec can be simulated using the private key of the encryption scheme and the public key of the signature scheme.

Finally, the adversary $\mathcal{A}$ returns (or asks to VerDec) a new *signed–ciphertext* (c_1, c_2) which is valid with probability ν. With the public key of the signature scheme, one can extract the message s_2 signed into c_2. By definition, (s_2, c_2) is an existential forgery for the signature scheme. Indeed, one has just to check whether it is a really new signed-message. Because of the determinism of the sharing technique, after the random choice of r, if s_2 has already been signed by the oracle SigEnc, it was for encrypting and signing $m\|r$, where

$$s_2 = (m\|r) + 2h(m\|r) \bmod p,$$

which is uniquely defined in the list of the queries asked to the random oracle h, unless one has found a collision for the function

$$G : x \mapsto x + 2h(x) \bmod p,$$

between the q_1 values given by the simulation and the q_H answers obtained by the adversary. Because of the randomness of the random oracle h, this is upper-bounded by $q_1 \cdot q_H/2^k$.

Furthermore, one has to be sure that everything looks like in a real attack, from the adversary $\mathcal{A}$ point of view. However, when one defines a value for h, it may have already been defined (event BadH). The probability of such an event is less than $q_H/2^{k_2}$, for each simulation of the oracle SigEnc, because of the randomness of r.

Finally, the probability for $\mathcal{B}$ to produce an existential forgery against the signature scheme is greater than

$$\nu - q_1 \cdot q_H \times \left(\frac{1}{2^{k_2}} + \frac{1}{2^k}\right).$$

Furthermore, one can easily see that $q_H \leq q_h + q_1 + q_2$: the probability of the machine for producing and existential forgery is greater than

$$\mathsf{Succ}^{\mathsf{nef-rma}}_{\mathsf{Sign}}(\mathcal{B}) \geq \mathsf{Succ}^{\mathsf{nef-ada}}_{\mathsf{SignCrypt}}(\mathcal{A}) - q_1 \cdot (q_1 + q_2 + q_h) \times \left(\frac{1}{2^{k_2}} + \frac{1}{2^k}\right).$$