Efficiency and Security Gains via Randomness Reuse Across Different Cryptographic Primitives

Author: Afonso Delerue Arriaga
Supervisor: Prof. Manuel Bernardo Barbosa

January, 2011
University of Minho

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. Manuel Barbosa, for his guidance and persistence. The enthusiasm he shows for modern cryptography, his knowledge and teaching skills, contributed considerably to my research experience. Thank you!

I also wish to thank Dr. Pooya Farshim for receiving me at Royal Holloway, University of London. He has done a thorough reading of my dissertation, which generated many recommendations and suggestions that were invaluable for my research. He also arranged several meetings with students of his faculty whose research is related to mine to exchange ideas, which have proved very constructive.

I would also like to thank my parents for the support they provided me through my entire life, without whose endless encouragement, I would not have finished this dissertation.

Finally, I must acknowledge that my work was supported by the *Fundação para a Ciência e a Tecnologia* (FCT) under grant UMINHO/BI/158/2009.

Abstract

The use of random coins in the implementation of cryptographic algorithms is needed in order to achieve higher security levels. This can however be costly in terms of bandwidth and computation. Minimizing the amount of fresh randomness required is important for an overall efficiency but could also lead to security flaws. Therefore, randomness reuse is an optimization that must be carefully studied on a theoretical level in order to attain the necessary security.

In this dissertation we look at how randomness reuse can be applied across different cryptographic primitives. We focus our attention on joint signature and encryption, as well as on the KEM-DEM paradigm, where randomness reuse can, in specific circumstances, provide not only efficiency gains, but also additional security guarantees.

Resumo

A utilização de aleatoriedade na implementação de algoritmos criptográficos é necessária para satisfazer requisitos de segurança mais exigentes. Esta operação pode, no entanto, revelar-se dispendiosa a nível de largura de banda e de esforço computacional. Minimizar o recurso à aleatoriedade é importante para melhorar a eficiência dos algoritmos, embora eventuais falhas de segurança inerentes ao processo não devam ser desconsideradas. A reutilização de aleatoriedade entre primitivas criptográficas é portanto uma optimização que deve ser cuidadosamente estudada a nível teórico para que se possa alcançar os padrões de segurança desejados.

Nesta dissertação, procura-se estudar a reutilização de aleatoriedade entre diversas primitivas criptográficas, dando ênfase à sua aplicação entre as primitivas criptográficas de assinatura e de cifra, assim como ao paradigma KEM-DEM. Resultam desta técnica não só ganhos de eficiência, como também, em casos particulares, novas garantias de segurança.

Contents

1 Introduction **1**

1.1 Our contribution 2

1.2 Dissertation organisation 3

2 Related Work **4**

2.1 The role of definitions in public-key cryptography 4

2.2 Game hopping proofs 5

2.3 Computational security 8

2.4 Standard model vs. random oracle model 9

2.5 Randomness reuse in multi-recipient encryption schemes 11

2.6 Signcrytion, one primitive for authenticity and confidentiality 11

2.7 Hybrid encryption 12

3 Preliminaries **14**

3.1 Notation 14

3.2 Public-key encryption 15

3.3 Digital signature 16

3.4 Signcryption 18

3.4.1 Insider and outsider adversaries 19

3.5 KEM-DEM 21

3.5.1 Key encapsulation mechanism 21

3.5.2 Data encapsulation mechanism 22

3.5.3 Hybrid construction 23

4 Security of Sequential Compositions of Signature and Encryption **25**

4.1 Two constructions 25

4.2 gCCA2-security, a relaxation of CCA2-security 26

4.3 From a two-user setting to a multi-user setting 27

4.4 Security proofs 28
4.5 Results summary 28

5 Security of Sequential Compositions of Signature and Encryption with Randomness Reuse 30
5.1 Partitioned and compatible schemes 30
5.2 Two constructions with randomness reuse 31
5.3 Reproducibility 32
5.4 Randomness dependent attacks 33
5.4.1 Multiple encryption queries 34
5.5 New theorems and proofs 38
5.6 Results summary 52
5.7 Candidates for instantiation 52

6 KEM-DEM with Randomness Reuse 54
6.1 The construction 54
6.2 Reproducibility 55
6.3 New theorem and proof 55
6.4 Possible instantiations 58

7 Conclusions and Future Directions 60

Appendices 62

Bibliography 70

Chapter 1

Introduction

Modern cryptography deals with much more than just hiding information; mathematicians and computer scientists study methods to securely (and efficiently) transmit messages through computer networks. Depending on the intended goal, secure transmissions aim to assure confidentiality on the transmitted messages, or the messages' authenticity and integrity. On many applications, both guarantees are needed.

Confidentiality is achieved thanks to encryption schemes. The security of such schemes can be defined in several ways. Intuitively, one may argue that an encryption scheme should hide a message in such a way that any part of the message becomes unreadable to anyone but the intended receiver. To formalize this intuition, it is required that an encryption scheme produces ciphertexts (from messages of the same size) that are indistinguishable from each other. Loosely speaking, the ciphertexts should look-alike to any illegitimate user, in the sense that given any two messages and a ciphertext of one of those messages, an illegitimate user should not be able to tell to which message the ciphertext corresponds to[1]. This notion of security applies to both symmetric and asymmetric encryption schemes.

Although asymmetric cryptography simplifies key distribution, symmetric cryptosystems are much more efficient. Hence, in the public-key setting, hybrid encryption schemes take advantage of keyed symmetric algorithms. Cramer and Shoup [36, 16] developed a framework where hybrid encryption schemes are constructed with two distinct components: an asymmetric key encapsulation mechanism (KEM) and a symmetric data encapsulation mechanism (DEM).

Signature schemes have the purpose to mimic handwritten signatures. They offer authenticity and integrity; a digital signature over a message binds the signer to the

[1] A precise definition of ciphertext-indistinguishability is given in Section 3.2

message. Because both signature and encryption schemes are two fundamental cryptographic primitives, they can be joined together to form a new primitive called signcryption [5], which fulfills the functionalities of both schemes.
Randomness is employed in several cryptographic schemes in order to reach strong security definitions such as indistinguishability of ciphertexts, and to ease the construction of signature schemes. However, the use of random coins comes with an additional computational cost and possibly a higher bandwidth usage. The generation of the randomness itself, as well as the operation usually needed to hide the random coins from potential attackers, are resource consuming. This is more noticeable as the size of fresh randomness increases, which is the case when the same algorithm is instantiated a number of times, or different algorithms are used. One approach to minimize this problem is to reuse randomness across the multiple instantiations of algorithms, previously done in the context of batch operations where messages are encrypted to multiple recipients [8]. It could be even more challenging to reuse randomness across *different* cryptographic primitives.
New scenarios where randomness reuse could be employed are the KEM-DEM paradigm, or between a signature scheme and an encryption scheme. Despite the gain in load-processing and bandwidth, this optimization must be pursued with caution as it can lead to security flaws.

1.1 Our contribution

The aim of this dissertation is to extend the existing results on randomness reuse and study its application across different primitives. We focus our attention on the combination of signature and encryption, where randomness reuse can, in specific circumstances, provide not only efficiency gains, but also *additional* security guarantees. This includes finding the proper definition of security for each case, by describing the security goals and the attack models, and presenting the required properties for identifying wherever randomness reuse can be safely employed. The benefits and disadvantages of randomness reuse are also highlighted.
We closely follow the results from [5], where the requirements for generic constructions of signcryption primitives are stated. The motivation for this work is to generalize the optimization techniques typically used in the construction of concrete signcryption schemes [42].
Then, we turn our attention to the KEM-DEM paradigm. We show that this combi-

nation can be easily and naturally extended to allow for randomness reuse between the two components. We demonstrate this with the practical case of Cramer-Shoup KEM [14], and one-time symmetric encryption based on the cipher-block chaining (CBC) mode of operation DEM [20]. The advantage of this approach in comparison to common practice is that the initialization vector (IV) no longer needs to be derived from the common secret, which permits aligning the practical use of the KEM-DEM paradigm with existing theoretical results without any efficiency loss.

1.2 Dissertation organisation

We start in Chapter 2 by briefly discussing related work, security definitions, standard techniques employed in theoretic security proofs, and how to compare results in provable security.

In Chapter 3, we explain the notation used throughout this dissertation. We also give definitions for the various primitives used later on, including their syntax, attack models, and security goals. Since we focus on signcryption and KEM-DEM construction, we review some important aspects such as signcryption insider attacks and hybrid public-key encryption.

Chapter 4 revisits the results obtained by An, Dodis and Rabin [5] on sequential black-box compositions of signature and encryption schemes. Proofs from [5] are included in Appendix A so that they can easily be compared to the proofs of our theorems (the notation and code-based game playing technique employed are consistent with the rest of this document). We then extend these results in Chapter 5 to apply the same randomness across signature and encryption schemes. Finally we compare our results to those already known, and show candidates that may satisfy the requirements for instantiation of this framework.

KEM-DEM receives the same treatment in Chapter 6. Randomness reuse is applied across KEM and DEM, and a possible instantiation for this construction is presented.

Finally, in Chapter 7 we discuss the relevance and implications of the results in this dissertation.

Chapter 2

Related Work

2.1 The role of definitions in public-key cryptography

Until 1982, before the work of Goldwasser and Micali, who introduced the notion of security proof for a public-key encryption scheme [25], there could be no assurance that schemes were secure beyond resistance to specific attacks. An encryption scheme satisfying the definition of security introduced by Goldwasser and Micali is known as semantically secure, a widely-used definition of security for asymmetric encryption schemes. For an encryption scheme to be semantically secure, it must be infeasible for a computationally-bounded adversary to extract any information (except the size of the message) from a ciphertext, even when the encryption algorithm and the public-key are known. The primary motivation of this seminal was to propose an encryption scheme provably secure under a complexity-theoretic assumption. The proof took the form of a reduction: if the algorithm didn't meet the security definition, the quadric residuosity problem wouldn't be intractable. This marks the beginning of provable security. From a point of view shared by many authors, beyond being an art, cryptography became a science [34, 28].

Semantic security is equivalent to ciphertext-indistinguishability under chosen plaintext attacks [26] (also introduced by Goldwasser and Micali), which is a much more common definition nowadays because it better suits the constructions of security proofs. The interaction between the adversary and the encryption scheme made clear the need for separately defining the attack model and the security goal. The attack model specifies the capabilities of the adversary (e.g. which algorithms it can access during his attack), and the goal states when the adversary is deemed successful (e.g. it must tell which

of two messages are encrypted in the challenge-ciphertext). Together, they define the security model.
Later, other attack models emerged, and stronger capabilities are now given to the adversary. These new powers allow the adversary to decrypt ciphertexts of his choice; the models are called chosen-chipertext attacks. In the real world, an attacker may have access to some ciphertexts and the messages underneath (it could simply be the case that a confidential message becomes intentionally public, or some drafts were left behind). Even in this scenario, we would like the security of other encrypted messages not to be compromised. First, Naor and Yung [32] formulated non-adaptive chosen-chipertext attacks (CCA1), and later, Rackoff and Simon [33] strengtten this model to adaptive chosen-chipertext attacks (CCA2). In these models, the adversary is allowed to decrypt before the challenge is issued, or at any time except, for obvious reasons, the given challenge-ciphertext (CCA1 and CCA2 attack models, respectively). Although these attack models may seem to give unrealistic powers to the adversary (the adversary may choose which ciphertexts he wishes to decrypt), pessimistic definitions are better than definitions that may underestimate the adversary's capabilities. Still, several schemes satisfying these stronger definitions were found [23, 14], and CCA2 security is now considered a requirement for general-purpose encryption schemes.
Formal definitions of indistinguishability (IND) under chosen-plaintext attack (CPA), non-adaptive chosen-ciphertext attack (CCA1) and adaptive chosen-ciphertext attack (CCA2) models are given in Section 3.2. Analogous models for signature schemes exist and are presented in Section 3.3. Naturally, the security goal is no longer to distinguish between ciphertexts, but rather to forge signatures.

2.2 Game hopping proofs

Security proofs written in a conventional probabilistic language can be very complex and hardly verifiable. For this reason, proofs are not always error-free [29]. A technique popularized by Shoup [37], known as game-hopping, allows cryptographers to write proofs broken down into small steps.
In game-hopping proofs, we conceptualize the adversary's interaction with its environment as a game, which in the proof is part of a sequence of games. The first game is a transposition of the security model for the scheme we want to prove secure. So, we want to determine an upper-bound of the probability that an adversary has in winning

this game. If this upper-bound probability is very low[1], then the scheme is secure. The first game then suffers a small change, provably imperceptible to the adversary, resulting in a new game (this transition is called a game-hop). Other consecutive changes are applied until we obtain a game which is simple enough to calculate an upper-bound of the probability that an adversary has in winning it (see Figure 2.1). As a result, the adversary's probability of winning the first game is upper-bounded by the increased probability of success that comes from each game-hop, and the probability of winning the last game of the sequence. Because the probability of breaking a scheme is related to the probability of winning the game which is a transposition of the security model, we obtain an upper-bound of the chances a real attacker has in breaking the scheme.

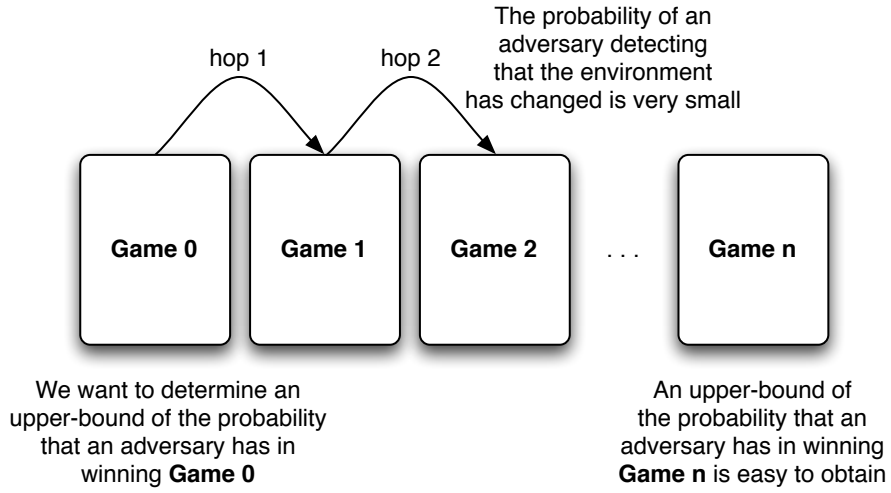

Figure 2.1: Security proof as a sequence of games

Adopting the guidelines given by Bellare and Rogaway [10], we define a game as a program that runs inside it an adversary, which is a sub-program. By describing games and adversaries in a pseudocode language, we make proofs more rigorous, as well as more easily verifiable. This technique also allows proofs to be checked by automatic verification tools [11], which is another upside.

The game has two environment procedures: Initialize and Finalize. All other procedures are oracles available to the adversary. At the beginning of the game, Initialize runs and its output is passed to the adversary. After receiving the output of Initialize, the adversary executes. During its execution, it may call any available oracle as it pleases. When the adversary terminates its execution, its output is passed to the Finalize procedure and the outcome of the game is known. Either the adversary won

[1]We define what "very low" means in Section 2.3.

the game or it lost the game. For a visual description of this interaction, see Figure 2.2 (this diagram is similar to Figure 2 of [10]).

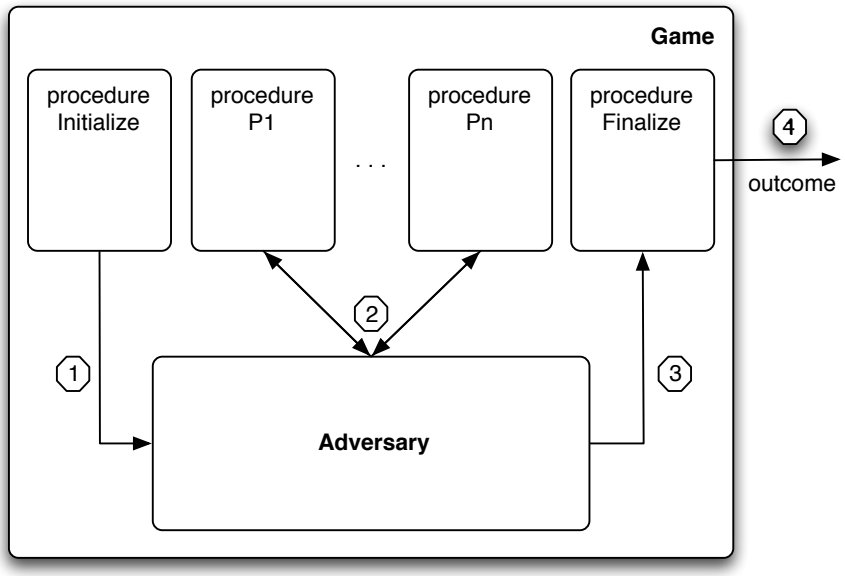

Figure 2.2: The interaction between an adversary an its environment conceptualized as a game

There are three types of game-hops [19]:

- **Bridging steps**
  The environment is rephrased but from the adversary's point of view nothing actually changes. The probability of the adversary winning the rephrased game is exactly the same as winning the previous game.

- **Transitions based on indistinguishability**
  The output of one of the procedure changes. It must be provable that the probability of winning the new game does not increase more than a very small amount. This can be proved by contradiction: if the adversary's behavior changes significantly, then we can build another adversary that uses the first one to do a computationally hard task (e.g. distinguish between values sampled from indistinguishable distributions).

- **Transitions based on failure events**
  A failure event is an event that occurs with a arguably very small probability. The new game is exactly as the previous one, except when this event occurs. From the Difference Lemma we conclude that the adversary's increased probability of

success from one game to another is not larger than the probability of the failure event occurring. By building an adversary that does a computationally hard task whenever the failure event occurs (e.g. solve a computationally hard problem), we upper-bound the probability of this incident.

Difference Lemma. *Let A,B,F be events defined in some probability distribution, and suppose that* $A \wedge \neg F \Leftrightarrow B \wedge \neg F$. *Then* $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.

Proof. The calculation is straightforward. We have

$$\begin{aligned} |\Pr[A] - \Pr[B]| &= |\Pr[A \wedge F] + \Pr[A \wedge \neg F] - \Pr[B \wedge F] - \Pr[B \wedge \neg F]| \\ &= |\Pr[A \wedge F] - \Pr[B \wedge F]| \\ &\leq \Pr[F]. \end{aligned}$$

□

Because code-based proofs are written in a pseudocode language, game-hops based on indistinguishability and failure events are argued by building adversaries in the same pseudocode language. All proofs in this dissertation use this code-based game-hopping technique. The notation employed is detailed in Section 3.1.

2.3 Computational security

Computational security can be defined for all primitives (symmetric encryption, public-key encryption, digital signature, ...). Contrarily to perfectly secure encryption [35] (which is impractical for most applications because it requires a key as long as all messages encrypted with it), computationally secure encryption can always be broken if unlimited resources were available to an attacker. However, no realistic attacker has unlimited computational power!

There are two common approaches to treat computational security: the asymptotic approach and the concrete security approach. The asymptotic approach relies on the security parameter of the scheme[2], and defines attackers as probabilistic polynomial-time (PPT) adversaries. An algorithm is probabilistic if it tosses some coins during its execution; several executions with the same input may result in different outputs.

[2]The security parameter is usually the key length.

It is said to run in polynomial time if there exists a polynomial $p(.)$ such that for every input $x \in \{0,1\}^*$, the algorithm runs at most within time $p(|x|)$, where $|x|$ is the size of the input. Here, time can be measured in steps, and the size of the input is that of the security parameter. Moreover, this approach equates the notion of "very small" with that of a function that decreases faster than any inverse polynomial in the security parameter of the scheme. A function with this behavior is called negligible (see Definition 2.1). Thus, a scheme is asymptotically secure if any PPT adversary has negligible probability of breaking the scheme. This approach guarantees the security of the scheme for sufficiently large values of the security parameter.

Definition 2.1. *A function $f(.)$ is negligible if for every polynomial $p(.)$ there exists an n such that for all integers $i > n$ it holds that $f(i) < \frac{1}{p(i)}$.*

While asymptotic security is an important guarantee, it fails to answer with which value the security parameter should be instantiated. This obviously depends on the attacker's computational power and how small we want his chances in breaking the scheme to be. The concrete security approach explicitly quantifies the maximum acceptable probability of success of any adversary running within a specific time. That is, let t be the amount of time an adversary is allowed to run (e.g. 2^{80} steps) and ϵ the maximum acceptable probability of an adversary breaking the scheme (e.g. 2^{-30}). Using the concrete security approach, a scheme is (t, ϵ)-secure if any adversary running for no longer than t steps succeeds in breaking the scheme with probability at most ϵ. We skip the concrete security approach in this dissertation because the task of determining the security parameter depends on specific instantiations of our constructions and their usage.

2.4 Standard model vs. random oracle model

Some cryptographic schemes cannot be proven secure without making strong assumptions regarding the behavior of hash functions. For this reason, a popular methodology for designing cryptographic schemes models hash functions has random oracles. These oracles are a mathematical abstraction that map every input query to a point of the hash function's codomain, chosen uniformly at random, with one exception: if queried twice on the same input, the random oracle replies the same answer to both queries.

Schemes proven secure using this *idealized* model are described as being secure in the random oracle model, as opposed to secure in the standard model.

When comparing the current state-of-the-art schemes proven secure in the standard model to those proven secure in the random oracle model, it is clear that the latter are more efficient. The encryption scheme designed by Kurosawa and Desmedt [30], which encrypts computing one exponentiation less than the scheme of Cramer and Shoup [14] (both proven secure in the standard model), is still less efficient than the RSA-OAEP [9] scheme proposed by Bellare and Rogaway in 1994. On one hand, RSA-OAEP encrypts messages with only one exponentiation operation, while Kurosawa-Desmedt encryption algorithm executes four exponentiations; on the other hand, RSA-OAEP has only be proven secure in the random oracle model [23] while Cramer-Shoup and Kurosawa-Desmedt schemes just require a universal one-way hash function.

Looking at signature schemes, the scenario is not much different. Efficient signature schemes that have be proven secure in the standard model rely on stronger and less standard assumptions, such as Strong RSA[3] [15, 22, 24], or are based on bilinear maps [12, 40]. All these signature schemes have another important point in common: they are all *probabilistic*.

Although the random oracle model is a useful tool to provide *some* confidence about the security of a scheme, no "real" function can implement a true random oracle, and it is unclear that schemes proven secure in this model are indeed secure when the random oracle is replaced by a hash function such as SHA-256 [1]. In fact, Canetti, Goldreich and Halevi [13] created an artificial signature and encryption schemes proven secure in the random oracle model, but for which any implementation of the random oracle results in insecure schemes.

In this dissertation, we strictly rely on the standard model. However, because we deal with primitives as black boxes, their instantiations must be secure in the standard model as well, otherwise the result is subject to the random oracle model if one of the primitives is.

[3]Please refer to one of the articles for details about the Strong RSA assumption.

2.5 Randomness reuse in multi-recipient encryption schemes

Bellare, Boldyreva and Staddon [8] studied the problem of reusing randomness examining multi-recipient encryption, and consider the particular case of constructing such schemes by running multiple instances of a public-key encryption scheme, whilst sharing randomness across them. A suitable security definition was given and a general method, called reproducibility test, for identifying public-key encryption schemes that are secure when used in this scenario was presented.
Therefore, in order to efficiently (with computational and bandwidth savings) use an encryption scheme as the base of a secure multi-recipient encryption scheme, the encryption scheme must be reproducible. This condition states that given a ciphertext of any message under any public-key, it is possible to create a new ciphertext for a new message under an arbitrary key pair, such that the ciphertext uses the same random coins as that of the input ciphertext. This is a reasonable condition. Indeed, several popular encryption schemes have efficient reproduction algorithms.
This raises the following question: are there analogous reproducibility tests which detect when randomness can be safely reused across different primitives? We answer this question later on when we study the particular case of reusing random coins between signature and encryption schemes, as well as between the KEM and the DEM components of a hybrid encryption scheme.

2.6 Signcrytion, one primitive for authenticity and confidentiality

Signature and encryption are two fundamental cryptographic primitives. The former guarantees authenticity, integrity and non-repudiation. The latter achieves confidentiality. Because both primitives are often needed in secure communication, a single primitive called signcryption accomplishes the functionalities of signature and encryption in a single step. This primitive aims to simplify the usage of cryptographic schemes for practitioners, while effectively decreasing the computational costs and communication overheads in comparison to the usage of signature and encryption as two independent blocks.
The first signcryption scheme was introduced by Zheng in 1997 [42]. Since then, a

number of researchers have dedicated their time designing more efficient and secure schemes [43, 6, 38, 5, 41, 31, 18, 17].

When building new signcryption schemes, one may think of doing so by joining a signature scheme and an encryption scheme in an encrypt-then-sign ($\mathcal{E}t\mathcal{S}$) or sign-then-encrypt ($\mathcal{S}t\mathcal{E}$) manner. From a theoretical point of view, the security of the joined signature and encryption construction may not be as expected. An, Dodis and Rabin [5] extensively studied these constructions.

Some signcryption schemes were already designed to take advantage of randomness reuse, such as the scheme designed by Barbosa and Farshim [7]. Although the primary concern of the authors was not to extensively study randomness reuse in signcryption schemes, but rather present a specific construction. The proposed scheme can still be seen as an encrypt-then-sign construction with random coins shared between encryption and signature. It is interesting to note that this resulted in even stronger security[4] due to the extra binding provided by randomness reuse. Although the authors designed their scheme in the certificateless scenario, we restrict ourselves to the standard public-key setting.

In Chapter 5 we extend the generic constructions used in [5] to those reusing randomness across signature and encryption components, and take advantage of the benefits this optimization may offer.

2.7 Hybrid encryption

In symmetric-key cryptosystems, the sender and the receiver must share a common secret key in order to communicate securely. Although symmetric-key encryption schemes are usually very efficient, the same key is used to encrypt and decrypt the data (each two parties must agree on some secret key in order to establish a secure channel to communicate). In a multi-user setting, messages are exchanged between multiple parties and a different secret key is required for each channel.

In a multi-user scenario, public-key cryptosystems are more convenient in that each party possesses a private decryption key and a corresponding public encryption key. In order to send an encrypted message, the sender only needs to be aware of the receiver's public-key; no setup between the sender and the receiver is required. However, public-key encryption schemes usually rely on more complex computations, which leads to

[4]The scheme has full insider security (see Section 4.3).

lower efficiency. Moreover, "pure" public-key encryption schemes have a limited message space (which depends on the setup parameters of the scheme), while symmetric-key encryption schemes have an infinite message space.

Hybrid encryption schemes take advantage of both worlds: they have an unlimited message space, and are almost as efficient as symmetric-key encryption schemes (for long messages), while still in the public-key setting. Cramer and Shoup [36, 16] developed a framework where hybrid encryption schemes are constructed with two distinct components: a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM). The KEM, which is the public-key component, encapsulates a random secret-key encrypting it with the receiver's public-key. The DEM, which is a symmetric-key encryption scheme, encapsulates the data, encrypting it with the randomly chosen secret-key. The ciphertext is composed by both the key-encapsulation and the data-encapsulation mechanisms. The receiver can obviously recover the secret-key with his own private-key, and then decrypt the encrypted data with the secret-key. Note that a hybrid encryption scheme is itself a public-key encryption scheme (the public-key and the private-key are the same as in the KEM). A formal definition of this construction is given in Section 3.5.

Chapter 3

Preliminaries

3.1 Notation

Let $a, b, \ldots, Z$ be elements of $\{0,1\}^\star$ unless stated otherwise. Let $\mathsf{A}, \mathsf{B}, \ldots, \mathsf{Z}$ be algorithms, and $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$ sets. We write $a \leftarrow b$ to denote the algorithmic action of assigning the value of b to the variable a. We write $a \xleftarrow{\$} \mathcal{A}$ for sampling a from the set $\mathcal{A}$ uniformly at random. $\perp_1, \perp_2, \ldots, \perp_n \notin \{0,1\}^\star$ are special failure symbols. If A is a probabilistic algorithm we write $a \xleftarrow{\$} \mathsf{A}(i_1, i_2, \ldots, i_n)$ for the action of running A on inputs $i_1, i_2, \ldots, i_n$ with random coins, and assigning the result to a. Sometimes we run A on specific coins r and write $a \leftarrow \mathsf{A}(i_1, i_2, \ldots, i_n; r)$. For a space $\mathcal{S} \subseteq \{0,1\}^\star$, we identify $\mathcal{S}$ with its characteristic function. In other words, $f(a) = \mathsf{T}$ if and only if $a \in \mathcal{S}$. The symbol $\cup$ is used for the union of sets, e.g., $\mathcal{S} \leftarrow \mathcal{S} \cup \{a\}$ adds the element a to the set $\mathcal{S}$. The relative complement of $\mathcal{A}$ in $\mathcal{B}$ is denoted by $\mathcal{B} \setminus \mathcal{A}$. Therefore, if we want to remove the element a from the set $\mathcal{S}$, we simply write $\mathcal{S} \leftarrow \mathcal{S} \setminus \{a\}$.
When elements can be repeated or the their order matters, we consider lists as another data structure. The empty list is represented by square brackets []. $\mathsf{List} \leftarrow a : \mathsf{List}$ appends the element a to the head of List. The function $first()$ returns the first element of a list: $a \leftarrow first(\mathsf{List})$ assigns the value of the first element of List to the variable a. The function $tail()$ returns the list passed as argument without the first element, e.g., $\mathsf{List} \leftarrow tail(\mathsf{List})$ removes the first element from List.
'Find' is used for searching a set and assigning values to uninitialized elements with the first match that satisfies a certain condition. For exemplification purpose, suppose that the element a as no value assigned to it yet, and suppose that $\mathcal{S}$ is a set of pairs of integers. Find $(a, 3) \in \mathcal{S}$ If Not Found $a \leftarrow 0$ assigns to a the first element of the first pair found in $\mathcal{S}$ which as the value 3 in its second element. If no such pair is found,

the value 0 is assigned to a.
'If', 'Then', 'Else' are logic operators. The symbol | means "such that". Sometimes it is useful to include comments in the code. To do so, lines start with %, only exist for understanding purposes, and can be discarded. A gray background is used to highlight the changes in each game hop.

3.2 Public-key encryption

A public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is specified by three polynomial-time algorithms (in the length of their inputs) associated with a message space $\mathcal{M}$ and a randomness space $\mathcal{R}$.

- $\mathsf{Gen}(1^\lambda)$ is the probabilistic key-generation algorithm which takes as input the security parameter and returns a secret key sk and a matching public key pk. The security parameter is given as a string of 1's of size λ to be consistent with standard convention that requires all algorithms to run in polynomial time in the length of their input.

- $\mathsf{Enc}(m, pk; r)$ is the probabilistic encryption algorithm. On input a message $m \in \mathcal{M}$, a public key pk, and possibly some random coins $r \in \mathcal{R}$, this algorithm outputs a ciphertext c.

- $\mathsf{Dec}(c, sk)$ is the deterministic decryption algorithm. On input of a ciphertext c and a key sk, this algorithm outputs a message m or a special failure symbol $\perp$.

The correctness of a public-key encryption scheme requires that for any $\lambda \in \mathbb{N}$, any $(sk, pk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, any $m \in \mathcal{M}$, and any random coins $r \in \mathcal{R}$, we have that $\mathsf{Dec}(\mathsf{Enc}(m, pk; r), sk) = m$.
The standard notions of security for a public key encryption scheme are indistinguishability under various attack models. We define games for the most common attack models: chosen-plaintext attacks (Figure 3.1), chosen-chiphertext attacks (Figure 3.2) and adaptive chosen-ciphertext attacks (Figure 3.3). The games are played as described in Section 2.2, except that the adversary may only call Left-Right *once* with $m_0, m_1 \in \mathcal{M}$ such that $|m_0| = |m_1|$. The advantage of an adversary $\mathcal{A}$ against a public-key encryption scheme $\mathcal{E}$ is defined by $\mathbf{Adv}^{\mathsf{IND\text{-}ATK}}_{\mathcal{E}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}ATK}^{\mathcal{A}}_{\mathcal{E}} \Rightarrow \mathsf{T}] - 1$.

In game IND-CPA no decryption oracle is available to the adversary.

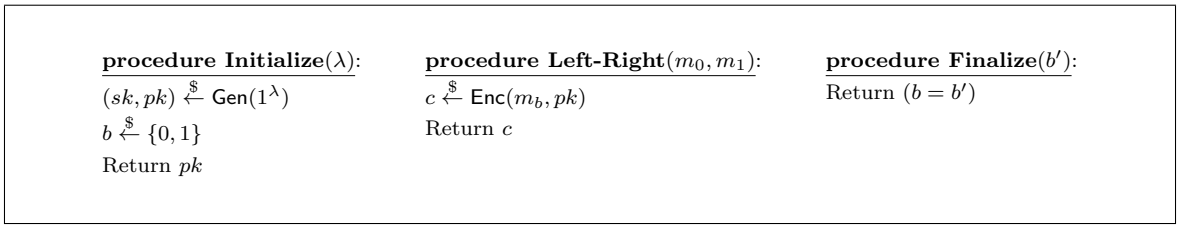

Figure 3.1: Game IND-CPA for a public-key encryption $\mathcal{E}$

Game IND-CCA1 has a decryption oracle, but the adversary may only query it before the challenge is issued (i.e., before querying Left-Right).

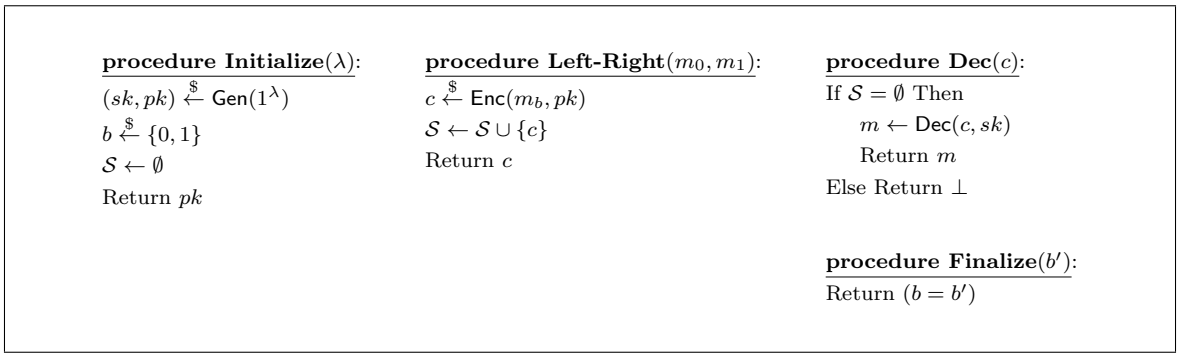

Figure 3.2: Game IND-CCA1 for a public-key encryption $\mathcal{E}$

In game IND-CCA2, the adversary may call the decryption oracle whenever he pleases but may *not* ask for the decryption of the challenge.

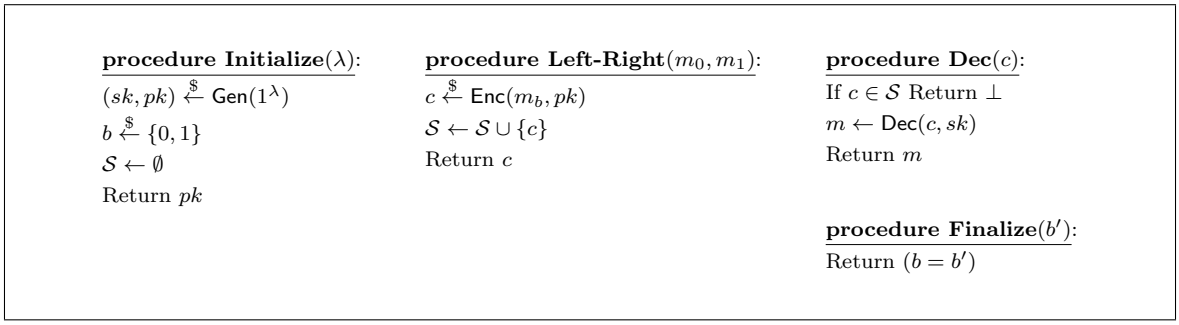

Figure 3.3: Game IND-CCA2 for a public-key encryption $\mathcal{E}$

3.3 Digital signature

A digital signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is specified by three polynomial-time algorithms (in the length of their inputs) associated with a randomness space $\mathcal{R}$ (we

assume that signature schemes are based on the hash-then-sign paradigm and so the message space $\mathcal{M} = \{0,1\}^\star$ is infinite).

- $\mathsf{Gen}(1^\lambda)$ is the probabilistic key-generation algorithm which takes as input the security parameter and returns a secret key sk and a matching public key pk.
- $\mathsf{Sign}(m, sk; r)$ is the probabilistic signature generation algorithm. On input a message m, a secret key sk, and possibly some random coins $r \in \mathcal{R}$, this algorithm outputs a signature σ.
- $\mathsf{Verify}(m, \sigma, pk)$ is the deterministic signature verification algorithm. On input of a signature σ, a message m and a public key pk, this algorithm outputs a boolean value T or F.

The correctness of a signature scheme requires that for any $\lambda \in \mathbb{N}$, any $(sk, pk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, any $m \in \{0,1\}^\star$, and any random coins $r \in \mathcal{R}$, we have $\mathsf{Verify}(\mathsf{Sign}(m, sk; r), m, pk) = \mathsf{T}$.

The standard notion of security for a digital signature scheme is unforgeability under chosen-message attacks at various strengths. Again, we define games for the most common attack models: existential unforgeability under no-message attacks (Figure 3.4), existential unforgeability under chosen-message attacks (Figure 3.5) and strong existential unforgeability under chosen-message attacks (Figure 3.6). The advantage of an adversary $\mathcal{A}$ against a digital signature scheme $\mathcal{S}$ is defined by following expression: $\mathbf{Adv}^{\mathsf{(s)UF\text{-}ATK}}_{\mathcal{S}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathsf{(s)UF\text{-}ATK}^{\mathcal{A}}_{\mathcal{S}} \Rightarrow \mathsf{T}]$.

Game $\mathsf{UF\text{-}NMA}$ has no signing oracle.

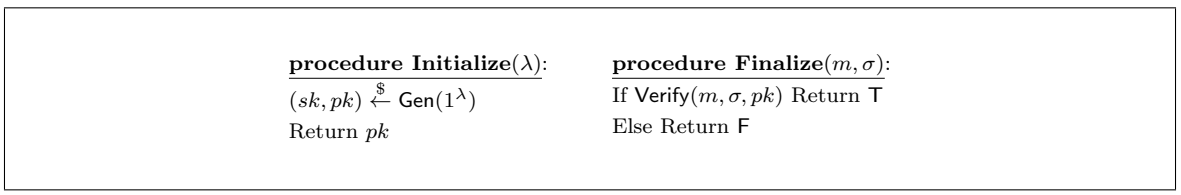

Figure 3.4: Game $\mathsf{UF\text{-}NMA}$ for a digital signature $\mathcal{S}$

A signing oracle is available in game $\mathsf{UF\text{-}CMA}$, but no signature on a message that has been queried to the signing oracle is a valid forgery.

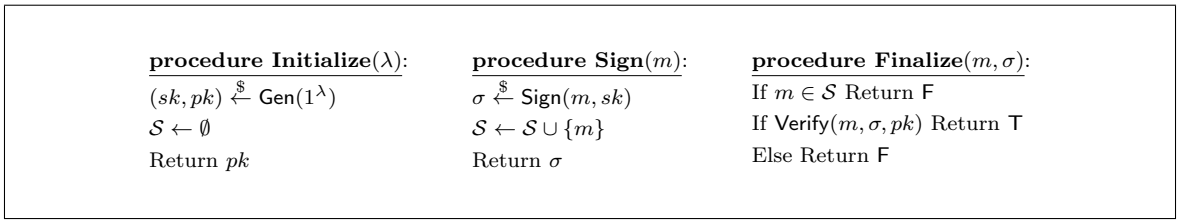

Figure 3.5: Game UF-CMA for a digital signature $\mathcal{S}$

In game sUF-CMA, only signatures returned by the signing oracle (with their corresponding messages) are invalid signatures.

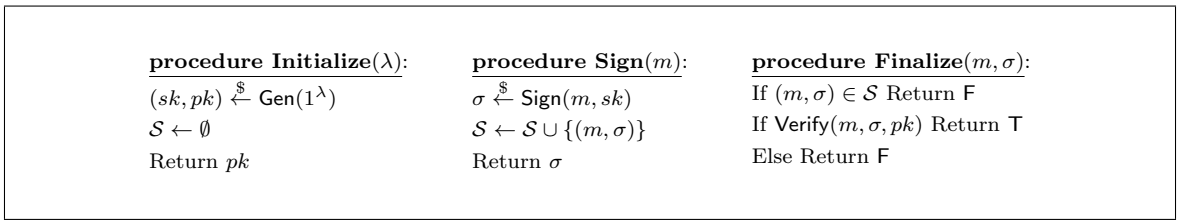

Figure 3.6: Game sUF-CMA for a digital signature $\mathcal{S}$

3.4 Signcryption

A signcryption scheme $\mathcal{SC} = (\mathsf{Gen}, \mathsf{Signcrypt}, \mathsf{Unsigncrypt})$ is specified by three polynomial-time algorithms (in the length of their inputs) associated with a message space $\mathcal{M}$ and a randomness space $\mathcal{R}$.

- $\mathsf{Gen}(1^\lambda)$ is the probabilistic key-generation algorithm which takes as input the security parameter and returns a secret key sk and a matching public key pk. Unless one wishes to signcrypt a message to oneself, two key pairs are required to signcrypt and unsigncrypt.
- $\mathsf{Signcrypt}(m, sk_S, pk_R; r)$ is the probabilistic signcryption algorithm. On input a message $m \in \mathcal{M}$, the secret key of the sender sk_S, the public key of the receiver pk_R, and possibly some random coins $r \in \mathcal{R}$, this algorithm outputs a signcryption ϕ.
- $\mathsf{Unsigncrypt}(\phi, pk_S, sk_R)$ is the deterministic unsigncryption algorithm. On input a signcryption ϕ, the public key of the sender pk_S, and the secret key of the receiver sk_R, this algorithm outputs a message m or a special failure symbol $\perp$.

The correctness of a signcryption scheme requires that for any $m \in \mathcal{M}$, any $\lambda \in \mathbb{N}$, any $(sk_S, pk_S) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, any $(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, and any random coins $r \in \mathcal{R}$,

we have $\mathsf{Unsigncrypt}(\mathsf{Signcrypt}(m, sk_S, pk_R; r), pk_S, sk_R) = m$.
Before defining games for the most common attack models as we did for public-key encryption and digital signature, we need to address some subtleties about signcryption. In the next section we look at two kind of adversaries that are relevant in this context, and we define confidentiality and authenticity games accordingly.

3.4.1 Insider and outsider adversaries

A signcryption scheme aims to protect the sender's authenticity and the receiver's confidentiality from everyone else. But, should this protection be extended to the actual agents involved in the communication? Suppose that Alice signcrypts a message to Bob. In one scenario, the adversary is an outsider adversary which only knows Alice and Bob's public keys. Another scenario captures a stronger notion of security in which Alice's authenticity is protected even against Bob, and Bob's confidentiality is protected even against Alice. This means that, if Alice "forgets" a message previously sent to Bob, she will no longer be able to learn anything about that message (this is known as forward secrecy, as it safeguards past messages from adversaries that corrupt Alice's secret key). Similarly, Bob should also be unable to create new ciphertexts that decrypt to valid messages under Alice's public key (this allows for non-repudiation to be achieved under some conditions). To model such scenarios, the adversary knows the sender's private key when playing a confidentiality game, and the receiver's private key when playing an authenticity game. These types of attackers are called insider adversaries.
We define here games against both types of adversaries, but only for the strongest notions of security. Games for other attack models can be easily written based on games in previous sections. Starting with confidentiality, we define games IND-oCCA2 and IND-iCCA2, which are played against outsider and insider adversaries, respectively. As before, Left-Right can only be called once. We define next sUF-oCMA and sUF-iCMA authenticity games. Again, these games are played against outsider and insider adversaries, respectively. The 'o' visibly stands for outsider adversary, and the 'i' for insider adversary. The advantage of an adversary $\mathcal{A}$ against the confidentiality of a signcryption scheme $\mathcal{SC}$ is defined by $\mathbf{Adv}^{\mathsf{IND\text{-}ATK}}_{\mathcal{SC}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}ATK}^{\mathcal{A}}_{\mathcal{SC}} \Rightarrow \mathsf{T}] - 1$.
Game IND-oCCA2 for signcryption is very similar to game IND-CCA2 for public-key encryption. However, notice the extra signcryption oracle which is needed because an outsider adversary cannot signcrypt messages by itself.

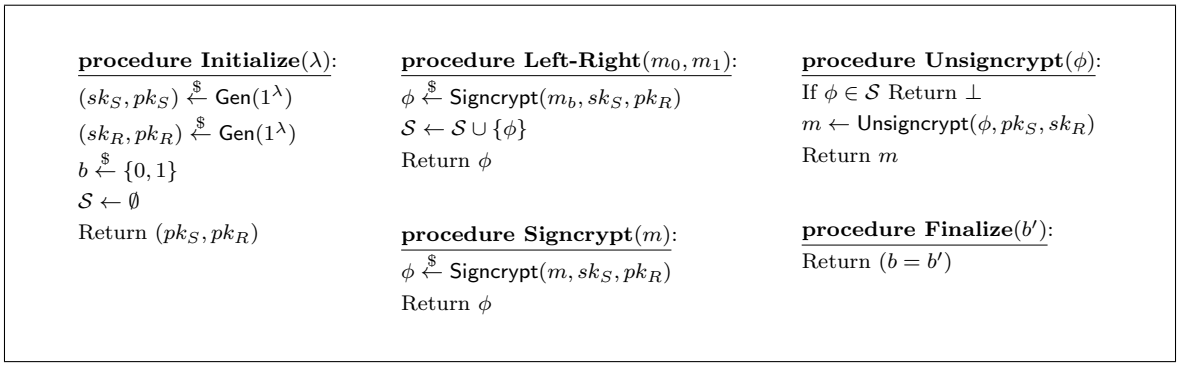

Figure 3.7: Game IND-oCCA2 for a signcryption $\mathcal{SC}$

In game IND-iCCA2, the signcryption oracle is no longer needed since the adversary now receives the sender's secret key.

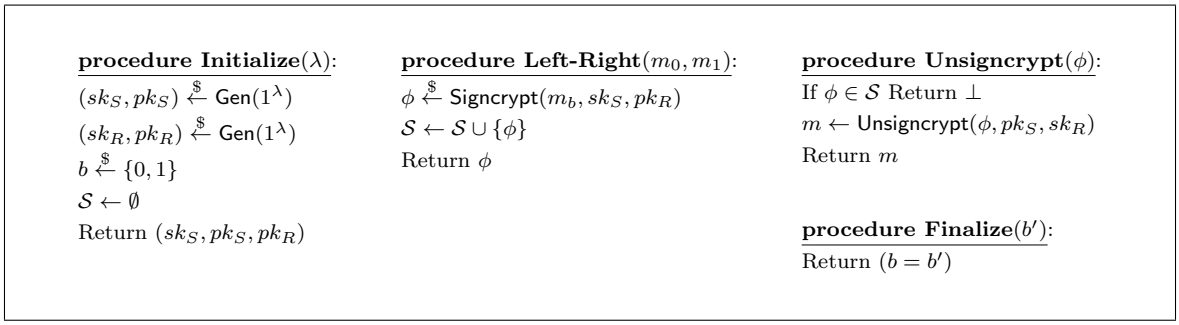

Figure 3.8: Game IND-iCCA2 for a signcryption $\mathcal{SC}$

The advantage of an adversary $\mathcal{B}$ against the authenticity of a signcryption scheme $\mathcal{SC}$ is defined by $\mathbf{Adv}^{\mathsf{(s)UF\text{-}ATK}}_{\mathcal{SC}}(\mathcal{B}) \stackrel{\text{def}}{=} \Pr[\mathsf{(s)UF\text{-}ATK}^{\mathcal{B}}_{\mathcal{SC}} \Rightarrow \mathsf{T}]$. Authenticity games are similar to those of a signature scheme, but in game sUF-oCMA, a unsigncryption oracle is available to outsider adversaries.

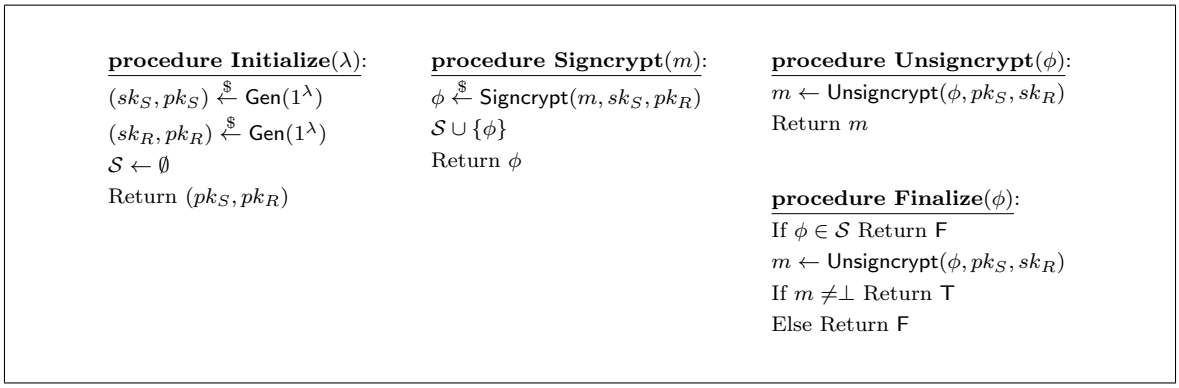

Figure 3.9: Game sUF-oCMA for a signcryption $\mathcal{SC}$

In game sUF-iCMA, the unsigncryption oracle is no longer necessary since the adversary now possesses the receiver's secret key.

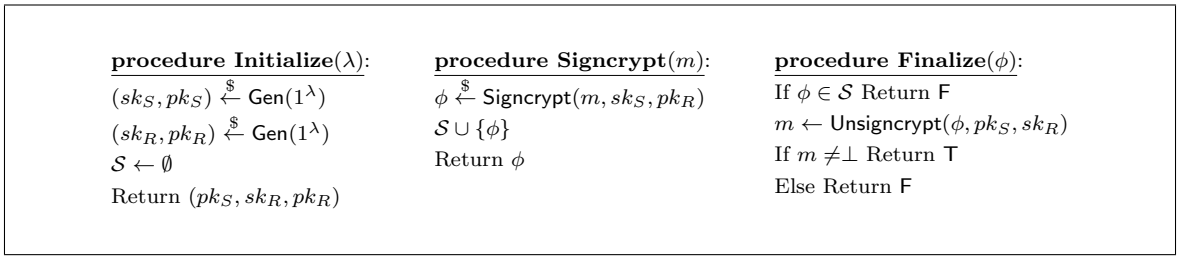

Figure 3.10: Game sUF-iCMA for a signcryption $\mathcal{SC}$

3.5 KEM-DEM

Cramer and Shoup developed a framework where hybrid encryption schemes are constructed with two distinct components: a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM). As stated in Section 2.7, a hybrid encryption scheme is itself a public-key encryption scheme. For this reason, the indistinguishability games are the same as for any other public-key encryption scheme. We first describe a KEM and a DEM separately, and then describe how a hybrid public-key encryption scheme can be constructed from the two components.

3.5.1 Key encapsulation mechanism

A key encapsulation mechanism $\mathcal{K} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is specified by three polynomial-time algorithms (in the length of their inputs) associated with a shared-key space $\mathcal{SK}$ and a randomness space $\mathcal{R}$, as follows.

- $\mathsf{Gen}(1^\lambda)$ is the probabilistic key-generation algorithm which takes as input the security parameter and returns a secret key sk and a matching public key pk.
- $\mathsf{Encap}(pk; r)$ is the probabilistic encapsulation algorithm. On input a public key pk, and possibly some random coins $r \in \mathcal{R}$, this algorithm outputs a pair containing a shared key $k \in \mathcal{SK}$, and a ciphertext c of k encrypted under pk.
- $\mathsf{Decap}(c, sk)$ is the deterministic decapsulation algorithm. On input a ciphertext c and a secret key sk, this algorithm outputs a shared key k or a special failure symbol $\perp$.

The correctness of a key encapsulation mechanism requires that for any $\lambda \in \mathbb{N}$, any $(sk, pk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$, and any random coins r, we have that $\mathsf{Encap}(pk; r) = (c, k) \Rightarrow \mathsf{Decap}(c, sk) = k$.
We only define here security against adaptive chosen-ciphertext attacks (Figure 3.11) since other attack models are not in the main focus of this work. The advantage of an adversary $\mathcal{A}$ against a key encapsulation mechanism $\mathcal{K}$ is defined by $\mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{K}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}CCA2}^{\mathcal{A}}_{\mathcal{K}} \Rightarrow \mathsf{T}] - 1$. In the following game, real-or-random oracle can only be called once.

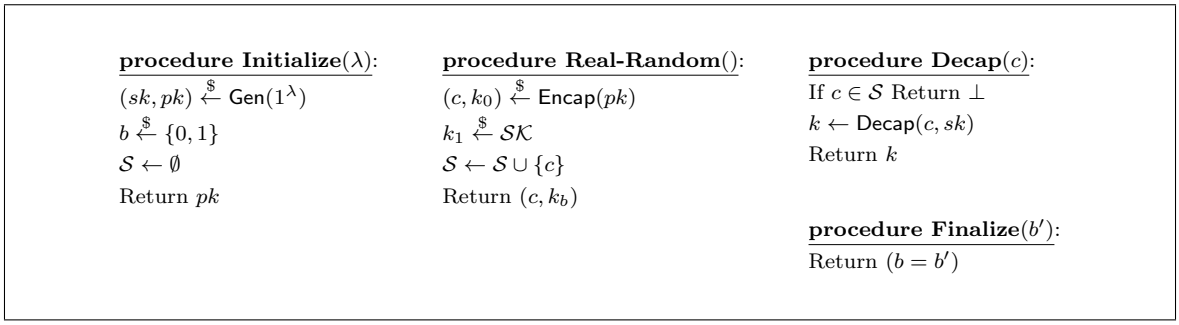

Figure 3.11: Game IND-CCA2 for a key encapsulation mechanism $\mathcal{K}$

3.5.2 Data encapsulation mechanism

A data encapsulation mechanism $\mathcal{D} = (\mathsf{Enc}, \mathsf{Dec})$ is specified by two polynomial-time algorithms (in the length of their inputs) associated with a key space $\mathcal{SK}$ and a randomness space $\mathcal{R}$.

- $\mathsf{Encap}(m, k; r)$ is the probabilistic encapsulation algorithm. On input a message $m \in \{0,1\}^\star$, a shared key $k \in \mathcal{SK}$, and possibly some random coins $r \in \mathcal{R}$, this algorithm outputs a ciphertext c.
- $\mathsf{Decap}(c, k)$ is the deterministic decapsulation algorithm. On input of a ciphertext c and a shared key $k \in \mathcal{SK}$, this algorithm outputs a message m or a special failure symbol $\perp$.

The correctness of a data encapsulation mechanism requires that for any $k \in \mathcal{SK}$, any $m \in \{0,1\}^\star$, and any random coins $r \in \mathcal{R}$ we have $\mathsf{Decap}(\mathsf{Encap}(m, k; r), k) = m$.
None of the security models for public-key encryption provides an encryption oracle because the adversary can produce ciphertexts of his choice by himself. To do so, he relies on the public-key made available in the beginning of the game. But this scenario does not apply to symmetric encryption. We now have to consider another dimension

for the definition of DEM's attack model: "single plaintext" or "multiple plaintext". The former model is adequate for one-time symmetric encryption schemes, since each shared key is only used to encrypt a single message; no encryption oracle is therefore available to the adversary in this model, and the only ciphertext the adversary obtains in the game is the challenge. For some applications, one must consider a "multiple plaintext" attack, where the adversary is allowed to obtain many encryptions of his choice, and not just a single encryption. In this attack model, beyond a decryption oracle, an encryption oracle is also available. However, a one-time symmetric encryption scheme is sufficient for the hybrid construction of Cramer and Shoup. Thus the security model defined here for DEM captures single plaintext and adaptive chosen-ciphertext attacks (Figure 3.12). The advantage of an adversary $\mathcal{A}$ against a data encapsulation mechanism $\mathcal{D}$ is defined by $\mathbf{Adv}_{\mathcal{D}}^{\mathsf{IND\text{-}CCA2}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}CCA2}_{\mathcal{D}}^{\mathcal{A}} \Rightarrow \mathsf{T}] - 1$. Once more, left-or-right oracle can only be called once.

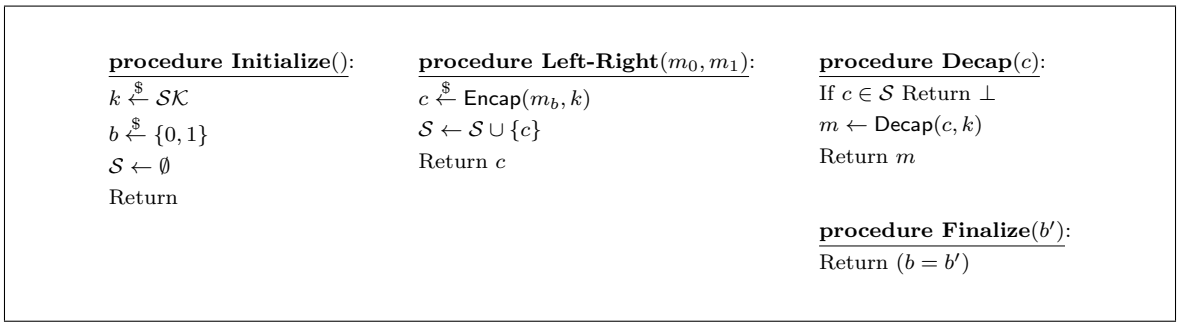

Figure 3.12: Game IND-CCA2 for a data encapsulation mechanism $\mathcal{D}$

3.5.3 Hybrid construction

First of all, the two components must be compatible: they must share the same key space $\mathcal{SK}$. The message space of a hybrid encryption scheme is that of the DEM, i.e., $\mathcal{M} = \{0,1\}^\star$. A hybrid encryption scheme $\mathcal{H} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is specified by three polynomial-time algorithms (in the length of their input) as follows:

- $\mathsf{Gen}(1^\lambda)$ is the same as $\mathcal{K}.\mathsf{Gen}(1^\lambda)$.
- $\mathsf{Enc}(m, pk)$ computes the sequence: $(c_1, k) \stackrel{\$}{\leftarrow} \mathcal{K}.\mathsf{Encap}(pk)$; $c_2 \stackrel{\$}{\leftarrow} \mathcal{D}.\mathsf{Encap}(m, k)$; $c \leftarrow (c_1, c_2)$; return c.
- $\mathsf{Dec}(c, sk)$ computes the sequence: $(c_1, c_2) \leftarrow c$; $k \leftarrow \mathcal{K}.\mathsf{Decap}(c_1, sk)$; $m \leftarrow \mathcal{D}.\mathsf{Decap}(c_2, k)$; return m.

Finally, a theorem from [16] concludes the security achieved with the above construction:

Theorem 1. *If KEM $\mathcal{K}$ and DEM $\mathcal{D}$ are secure against adaptive chosen ciphertext attacks, then so is the hybrid public-key encryption scheme $\mathcal{H}$.*

Chapter 4

Security of Sequential Compositions of Signature and Encryption

There are two obvious ways to transmit a message with authenticity and confidentiality: encrypt-then-sign ($\mathcal{E}t\mathcal{S}$) and sign-then-encrypt ($\mathcal{S}t\mathcal{E}$). Contrarily to what one may expect, this operation sometimes result in a *weaker* security than those assumed on encryption and signature components when analysed independently.

4.1 Two constructions

The constructions are very simple. Let $\mathcal{S}$ be a digital signature scheme and $\mathcal{E}$ a public-key encryption scheme. A signcryption scheme based on the encrypt-then-sign ($\mathcal{E}t\mathcal{S}$) construction works as follows:

- $\mathsf{Gen}(1^\lambda)$ runs $(sk_1, pk_1) \xleftarrow{\$} \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $(sk_2, pk_2) \xleftarrow{\$} \mathcal{E}.\mathsf{Gen}(1^\lambda)$, joins the keys together $(sk, pk) \leftarrow ((sk_1, sk_2), (pk_1, pk_2))$, and outputs (sk, pk).
- $\mathsf{Signcrypt}(m, sk_S, pk_R)$ computes the sequence: $(sk_1, sk_2) \leftarrow sk_S$; $(pk_1, pk_2) \leftarrow pk_R$; $c \xleftarrow{\$} \mathsf{Enc}(m, pk_2)$; $\sigma \xleftarrow{\$} \mathsf{Sign}(c, sk_1)$; $\phi \leftarrow (c, \sigma)$; return ϕ.
- $\mathsf{Unsigncrypt}(\phi, pk_S, sk_R)$ computes the sequence: $(pk_1, pk_2) \leftarrow pk_S$; $(sk_1, sk_2) \leftarrow sk_R$; $(c, \sigma) \leftarrow \phi$; if $\mathsf{Verify}(c, \sigma, pk_1)$ return $\mathsf{Dec}(c, sk_2)$ else return $\perp$.

A signcryption scheme based on the sign-then-encrypt construction ($\mathcal{S}t\mathcal{E}$) has the same Gen as above but the other algorithms differ:

- $\mathsf{Signcrypt}(m, sk_S, pk_R)$ computes the sequence: $(sk_1, sk_2) \leftarrow sk_S$; $(pk_1, pk_2) \leftarrow pk_R$; $\sigma \xleftarrow{\$} \mathsf{Sign}(m, sk_1)$; return $\mathsf{Enc}((m, \sigma), pk_2)$.

- $\mathsf{Unsigncrypt}(\phi, pk_S, sk_R)$ computes the sequence: $(pk_1, pk_2) \leftarrow pk_S$; $(sk_1, sk_2) \leftarrow sk_R$; $(m, \sigma) \leftarrow \mathsf{Dec}(\phi, sk_2)$; if $\mathsf{Verify}(m, \sigma, pk_1)$ return m else return $\perp$.

For simplicity of exposition, we assume that the message space of $\mathcal{E}$ is $\{0,1\}^\star$. Hence, the message space of $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ is also $\{0,1\}^\star$.

4.2 gCCA2-security, a relaxation of CCA2-security

Using a sequential composition of signature and encryption, security against insider attacks on both the strongest confidentiality and authenticity constrains (IND-iCCA2 and sUF-iCMA) cannot be attained. Briefly, this happens because an insider adversary can win the IND-iCCA2 game of an encrypt-then-sign construction by simply producing a new signature on the challenge ciphertext, and submitting the pair to the unsigncryption oracle. Analogously, an insider adversary can easily break the sign-then-encrypt construction by decrypting the ciphertext with its private-key and re-encrypting the result under the receiver's publick-key (this is a valid forgery in game sUF-iCMA).
An et al. [5] extensively studied these constructions, and noticed that the above constructions are secure when only UF-iCMA and IND-iCCA1 security are required. But, IND-iCCA1 is too weak of a definition to be adequate for a signcryption scheme when compared to UF-iCMA-security (which is a reasonable definition). To overcome this, they proposed a new attack model called *generalized* CCA2 (gCCA2).
An encryption scheme is secure against gCCA2 attacks if there exists an efficient decryption-respecting relation f with the property $f(c_1, c_2) = \mathsf{T} \Rightarrow \mathsf{Dec}(c_1) = \mathsf{Dec}(c_2)$ such that, besides preventing the adversary from querying the decryption oracle with ciphertexts satisfying the relation f with the challenge, the attack model is the same as CCA2. f is efficiently computable and captures cases of benign malleability, e.g., appending a bit to the ciphertext and querying decryption is no longer a successful attack. Formally, IND-gCCA2 is described in Figure 4.1.

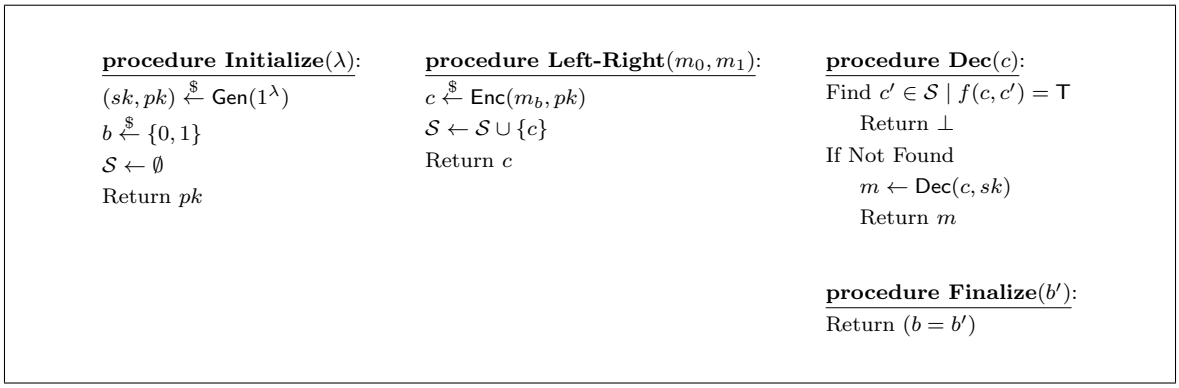

Figure 4.1: Game IND-gCCA2 for a public-key encryption $\mathcal{E}$

In this chapter, we revisit the main theorems in [5] which are based on this new security model. In Section 4.5 we synthesize and discuss the results of sequential compositions of signature and encryption.

4.3 From a two-user setting to a multi-user setting

Even UF-iCMA and IND-iCCA2 security fail to capture all possible attacks one can do to a signcryption scheme. Let ϕ be the signcryption of a message m Alice sent to Bob. If the signcryption scheme used by Alice is a simple sequential black box composition of encrypt-then-sign, Eve, who intercepted the signcryption ϕ, could simply discard the signature part of the signcryption ϕ, re-sign the ciphertext with her own private key, and convince Bob that she sent him the message m (without knowing m!). On the other hand, if the signcryption scheme is a sequential black box composition of sign-then-encrypt, Bob could decrypt the signcryption ϕ with his own private key, re-encrypt the message m and Alice's signature with Charlie's public key (let us call this other party Charlie), and convince Charlie that Alice sent him the message m.
An et al. [5] call these trivial attacks "identity fraud". The approach adopted by the authors is to study both constructions in the two-user setting, where these attacks are obviously not possible since no other user is considered, and then *fix* the construction to ensure security in the multi-user setting. By doing so, the constructions are simpler, and so are the security proofs.
The schemes resulting of $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$ constructions proven secure in the two-user settings remain secure in the multi-user setting, as long as the message is binded to the public key of the sender and to the public key of the receiver. So, the fix works as

follows:

- Before encrypting something, concatenate the public key of the sender (pk_S) to whatever is going to be encrypted.
- Before signing something, concatenate the public key of the receiver (pk_R) to whatever is going to be signed.
- The reverse process must validate that the public keys pk_S and pk_R match what is expected.

For a formal proof of the above transformation, please refer to [5].

4.4 Security proofs

The following theorems (theorems 2 and 3 of [5]) will serve as a starting point to our study. The proofs of these theorems are included in Appendix A for completeness. The notation described in Section 3.1 and the code-based game playing technique described in Section 2.2 are employed for consistency.

Theorem 2. *If* $\mathcal{E}$ *is* IND-CPA *and* $\mathcal{S}$ *is* UF-CMA*, then* $\mathcal{E}t\mathcal{S}$ *is* IND-ogCCA2[1] *and* UF-iCMA*.*

Theorem 3. *If* $\mathcal{E}$ *is* IND-gCCA2 *and* $\mathcal{S}$ *is* UF-NMA*, then* $\mathcal{S}t\mathcal{E}$ *is* IND-igCCA2[2] *and* UF-oCMA*.*

4.5 Results summary

Tables 4.1 and 4.2 summarise the generic results obtained by An et al. [5] for the $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ constructions. The authors argue that UF-CMA and IND-gCCA2 are the right definitions to assure authenticity and confidentiality. The cells in grey highlight the minimum conditions where the constructions satisfy these security definitions, against both insider and outsider adversaries.
For the reasons mentioned in Section 4.2, IND-iCCA2 does not appear in Table 4.1, as

[1] $\mathcal{E}t\mathcal{S}$ is IND-ogCCA2 with respect to the decryption-respecting relation f such that $f((c_1, \sigma_1), (c_2, \sigma_2)) = \mathsf{T} \Leftrightarrow c_1 = c_2$.
[2] $\mathcal{S}t\mathcal{E}$ is IND-igCCA2 with respect to the same decryption-respecting relation f of $\mathcal{E}$.

well as sUF-iCMA does not appear in Table 4.2. However, full outsider security[3] can be achieved with both $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ constructions.

Table 4.1: Results obtained by An et al. [5] for the $\mathcal{E}t\mathcal{S}$ construction.

	IND-CPA		IND-gCCA2		IND-CCA2	
UF-NMA	UF-oNMA	IND-oCPA	UF-oNMA	IND-ogCCA2	UF-oNMA	IND-ogCCA2
	UF-iNMA	IND-iCPA	UF-iNMA	IND-igCCA2	UF-iNMA	IND-igCCA2
UF-CMA	UF-oCMA	IND-ogCCA2	UF-oCMA	IND-ogCCA2	UF-oCMA	IND-ogCCA2
	UF-iCMA	IND-iCPA	UF-iCMA	IND-igCCA2	UF-iCMA	IND-igCCA2
sUF-CMA	sUF-oCMA	IND-oCCA2	sUF-oCMA	IND-oCCA2	sUF-oCMA	IND-oCCA2
	sUF-iCMA	IND-iCPA	sUF-iCMA	IND-igCCA2	sUF-iCMA	IND-igCCA2

Table 4.2: Results obtained by An et al. [5] for the $\mathcal{S}t\mathcal{E}$ construction.

	IND-CPA		IND-gCCA2		IND-CCA2	
UF-NMA	UF-oNMA	IND-oCPA	UF-oCMA	IND-ogCCA2	sUF-oCMA	IND-oCCA2
	UF-iNMA	IND-iCPA	UF-iNMA	IND-igCCA2	UF-iNMA	IND-iCCA2
UF-CMA	UF-oCMA	IND-oCPA	UF-oCMA	IND-ogCCA2	sUF-oCMA	IND-oCCA2
	UF-iCMA	IND-iCPA	UF-iCMA	IND-igCCA2	UF-iCMA	IND-iCCA2
sUF-CMA	UF-oCMA	IND-oCPA	UF-oCMA	IND-ogCCA2	sUF-oCMA	IND-oCCA2
	UF-iCMA	IND-iCPA	UF-iCMA	IND-igCCA2	UF-iCMA	IND-iCCA2

[3] "Full security" means security in terms of IND-CCA2 and sUF-CMA models.

Chapter 5

Security of Sequential Compositions of Signature and Encryption with Randomness Reuse

In this chapter we extend the results of Chapter 4 to construct signcryption schemes which use the same randomness across the two modular primitives. Our generic constructions capture concrete signcryption schemes as particular cases. Our results show that the joint use of signature and encryption is not necessarily a motivation for weaker security models that tolerate "benign" malleability (UF-iCMA and IND-gCCA2).
We propose a framework to construct signcryption schemes that are no longer subject to these malleability attacks, thanks to the additional binding provided by randomness reuse. We identify candidates that may fit in this framework, which guarantees full insider security (sUF-iCMA and IND-iCCA2).

5.1 Partitioned and compatible schemes

The notion of joint signature and encryption in the public-key setting with randomness reuse implies that the signature and encryption algorithms share the same random coins in their input. In order to clarify the concept and simplify the security proofs, we will restrict our attention to *partitioned* schemes that produce signatures and ciphertexts of the form (σ, R) and (c, R) respectively, where R depends only on the random coins r. Furthermore, we require the signature and encryption schemes to be *compatible*, i.e., on input the same random coins r, the signature and encryption algorithms produce

the same R. More precisely, a signature scheme $\mathcal{S}$ and an encryption scheme $\mathcal{E}$ are compatible if they share the same random space $\mathcal{R}$, and the code in Figure 5.1 always returns T for any message m.

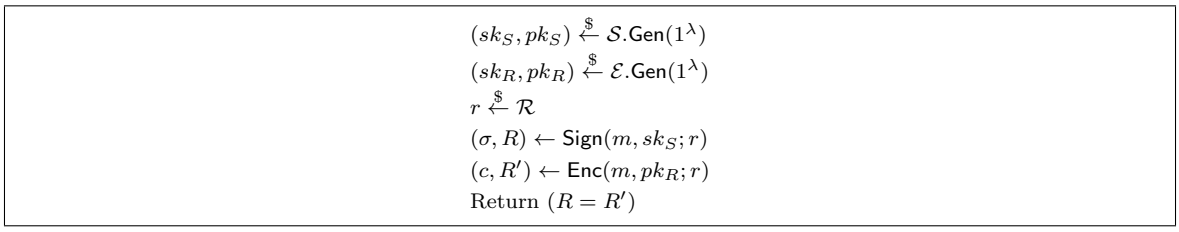

Figure 5.1: Compatibility between $\mathcal{S}$ and $\mathcal{E}$

Later in this dissertation we discuss instantiations for our constructions and show that partitioning is not an unnatural characteristic to impose on encryption and signature schemes.

5.2 Two constructions with randomness reuse

We briefly describe the constructions of Section 4.1 but now with randomness reuse. Let a digital signature $\mathcal{S}$ and a public-key encryption $\mathcal{E}$ be two *compatible* schemes, associated with the randomness space $\mathcal{R}$. A signcryption scheme based on the encrypt-then-sign construction ($\mathcal{E}t\mathcal{S}$) with randomness reuse works as follows:

- $\mathsf{Gen}(1^\lambda)$ runs $(sk_1, pk_1) \xleftarrow{\$} \mathcal{S}.\mathsf{Gen}(1^\lambda)$ and $(sk_2, pk_2) \xleftarrow{\$} \mathcal{E}.\mathsf{Gen}(1^\lambda)$, joins the keys together $(sk, pk) \leftarrow ((sk_1, sk_2), (pk_1, pk_2))$, and outputs (sk, pk).
- $\mathsf{Signcrypt}(m, sk_S, pk_R)$ computes the sequence: $(sk_1, sk_2) \leftarrow sk_S$; $(pk_1, pk_2) \leftarrow pk_R$; $r \xleftarrow{\$} \mathcal{R}$; $(c, R) \leftarrow \mathsf{Enc}(m, pk_2; r)$; $(\sigma, R) \leftarrow \mathsf{Sign}((c, R), sk_1; r)$; $\phi \leftarrow (c, \sigma, R)$; return ϕ.
- $\mathsf{Unsigncrypt}(\phi, pk_S, sk_R)$ computes the sequence: $(pk_1, pk_2) \leftarrow pk_S$; $(sk_1, sk_2) \leftarrow sk_R$; $(c, \sigma, R) \leftarrow \phi$; if $\mathsf{Verify}((c, R), (\sigma, R), pk_1)$ return $\mathsf{Dec}((c, R), sk_2)$ else return $\perp$.

In the sign-then-encrypt construction ($\mathcal{S}t\mathcal{E}$) with randomness reuse, Gen is the same as above but the other two algorithms differ:

- $\mathsf{Signcrypt}(m, sk_S, pk_R)$ computes the sequence: $(sk_1, sk_2) \leftarrow sk_S$; $(pk_1, pk_2) \leftarrow pk_R$; $r \xleftarrow{\$} \mathcal{R}$; $(\sigma, R) \leftarrow \mathsf{Sign}(m, sk_1; r)$; return $\mathsf{Enc}((m, (\sigma, R)), pk_2; r)$.

- $\mathsf{Unsigncrypt}(\phi, pk_S, sk_R)$ computes the sequence: $(pk_1, pk_2) \leftarrow pk_S$; $(sk_1, sk_2) \leftarrow sk_R$; $(c, R) \leftarrow \phi$; $(m, (\sigma, R')) \leftarrow \mathsf{Dec}((c, R), sk_2)$; if $R == R' \wedge \mathsf{Verify}(m, (\sigma, R), pk_1)$ return m else return $\perp$.

Once more, for simplicity of exposition we assume that the message space of $\mathcal{E}$ is $\{0,1\}^\star$. Comparing the size of the signcryption ϕ on both constructions, one notices that the $\mathcal{E}t\mathcal{S}$ construction produces a shorter output. This happens because both the ciphertext (c, R) and the signature (σ, R) are outputted, and since the R's are the same, only one needs to be included in the signcryption ϕ. In the $\mathcal{S}t\mathcal{E}$ construction, R is encrypted inside the ciphertext as well so that the unsigncryption algorithm can check for consistency. Intuitively this prevents an attacker from changing the encryption randomness, which is not authenticated by the signature in this case.

5.3 Reproducibility

Similarly to the requirements in [8], in order to provide simulations in our proofs, we need to define *reproduction* algorithms which achieve the same results as the standard signature and encryption algorithms but rely on R instead of r. To achieve this without contradicting security, these algorithms are also given the full key pair under which the ciphertext or signature is to be produced. We say that a signature scheme (resp. encryption scheme) is *reproducible* if there is a polynomial-time reproduction algorithm $\mathsf{Rep}_\mathcal{S}$ (resp. $\mathsf{Rep}_\mathcal{E}$) defined as follows:

- $\mathsf{Rep}_\mathcal{S}(m, sk_S, R)$ is a deterministic reproduction algorithm for signature. On input a message m, the signature key sk_S, and R, this algorithm outputs a signature σ[1] or a special failure symbol $\perp$.

- $\mathsf{Rep}_\mathcal{E}(m, sk_R, R)$ is a deterministic reproduction algorithm for encryption. On input a message m, the decryption key sk_R, and R, this algorithm outputs a ciphertext c[2] or a special failure symbol $\perp$.

The output of these reproduction algorithms must be consistent with their corresponding signature and encryption algorithms. Formally, reproducibility for $\mathcal{E}$ is defined in Figure 5.3 and reproducibility for $\mathcal{S}$ is defined in Figure 5.2. The schemes are reproducible if the code in these figures always returns T for any message m.

[1]The whole signature is in fact (σ, R), but R is given as input.
[2]The whole ciphertext is in fact (c, R), but R is given as input.

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$r \xleftarrow{\$} \mathcal{R}$
$(\sigma, R) \leftarrow \mathsf{Sign}(m, sk; r)$
$\sigma' \leftarrow \mathsf{Rep}_{\mathcal{S}}(m, sk, R)$
Return $(\sigma = \sigma')$

Figure 5.2: Reproducibility for $\mathcal{S}$

$(sk, pk) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$r \xleftarrow{\$} \mathcal{R}$
$(c, R) \leftarrow \mathsf{Enc}(m, pk; r)$
$c' \leftarrow \mathsf{Rep}_{\mathcal{E}}(m, sk, R)$
Return $(c = c')$

Figure 5.3: Reproducibility for $\mathcal{E}$

Notice that $\mathsf{Rep}_{\mathcal{E}}$ receives as input the secret (decryption) key, while Enc only takes the public (encryption) key. We show in Section 5.7 that reproducibility is available in many asymmetric encryption schemes.

5.4 Randomness dependent attacks

We introduce two new attack models: one for encryption and one for digital signatures. These new attack models are specific for partitioned schemes. We define a new security model for encryption, which we call indistinguishability under randomness dependent generalized chosen ciphertext attacks (IND-RDA).

As stated in Section 5.1, we restrict our attention to partitioned schemes which produce ciphertexts of the form (c, R), where R depends only on the random coins r, and encryption is otherwise deterministic if those coins are fixed. In Section 5.3 we introduced the notion of reproducibility. A reproduction algorithm is an algorithm that achieve the same result as the encryption algorithm of a scheme, but rely deterministically on R. So, it is clear that if we fix R, we are also fixing r, although we may not know its value.

This new IND-RDA model is similar to IND-gCCA2 except that the adversary receives R in the beginning of the game, before choosing the two messages for the challenger. For a correct simulation of this model, instead of splitting the encryption algorithm in order to obtain the subroutine which computes R from r, we simply encrypt any message[3] under random coins r, and save the pair (r, R). Also, the left-or-right oracle

[3]For the games presented in this section, we fixed the message to be "00000".

must only be queried once. The advantage of an adversary $\mathcal{A}$ against a partitioned public-key encryption $\mathcal{E}$ is defined by $\mathbf{Adv}^{\mathsf{IND\text{-}RDA}}_{\mathcal{E}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}RDA}^{\mathcal{A}}_{\mathcal{E}} \Rightarrow \mathsf{T}] - 1$. Game IND-RDA is defined in Figure 5.4.

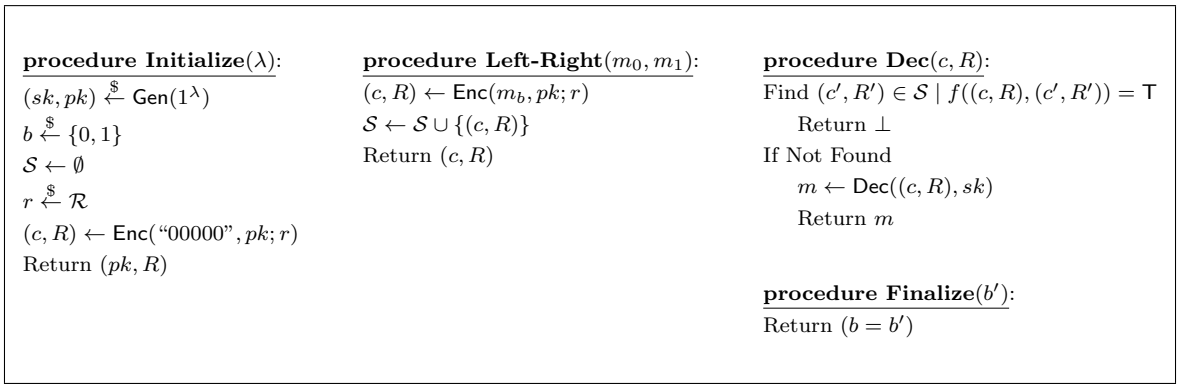

Figure 5.4: IND-RDA for a public-key encryption $\mathcal{E}$

We also introduce a new security model for signatures called unforgeability against randomness dependent chosen message attacks (UF-RDA). The observations made so far regarding IND-RDA apply analogously here. Here, the advantage of an adversary $\mathcal{A}$ against a partitioned signature scheme $\mathcal{S}$ is defined by $\mathbf{Adv}^{\mathsf{UF\text{-}RDA}}_{\mathcal{S}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[\mathsf{UF\text{-}RDA}^{\mathcal{A}}_{\mathcal{S}} \Rightarrow \mathsf{T}]$. Game UF-RDA is defined in Figure 5.5.

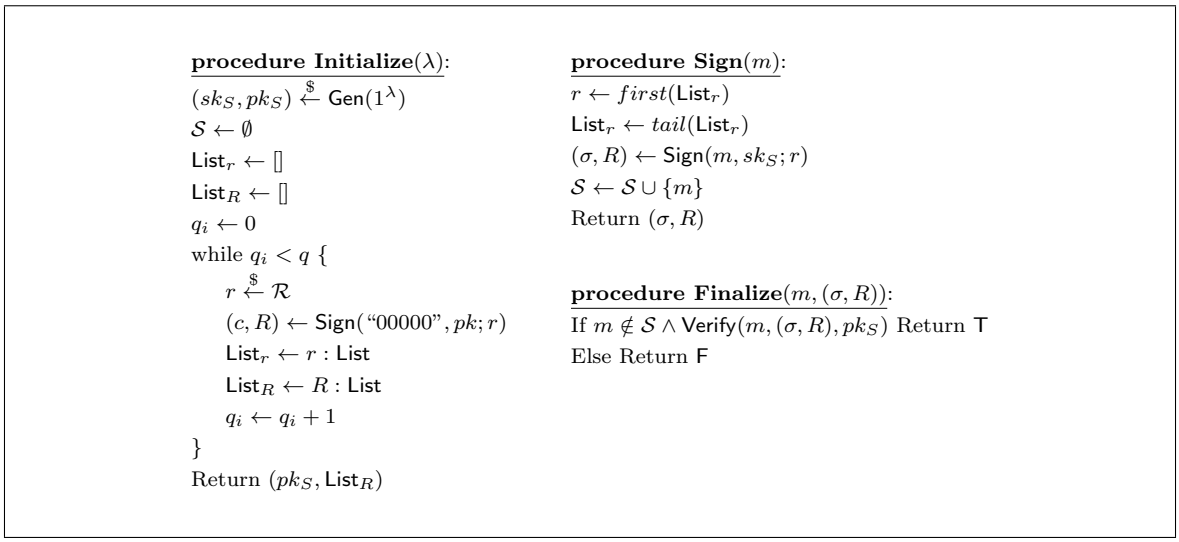

Figure 5.5: UF-RDA for a digital signature $\mathcal{S}$

5.4.1 Multiple encryption queries

In our previous definition of model IND-RDA, we restrict the adversary from making more than one query to the left-or-right oracle so that the model is as close as possible

to the model IND-gCCA2. However, we show here that no weakness is exposed in an IND-RDA encryption scheme via an attack involving the observation of ciphertexts of many related messages, chosen adaptively as a function of ciphertexts of previous messages. Although the adversary would gain a higher advantage in this scenario, the gain is very limited.

Let $\mathcal{B}$ be an adversary against IND-RDA-q (Figure 5.6), which is a game identical to IND-RDA except that now the adversary is given the capability of querying the left-or-right oracle at most q times. Furthermore, to be coherent with the new powers that allow the adversary to retrieve R before querying the left-or-right oracle, this game also outputs a list of length q of values for each R of each ciphertext to be outputted by this oracle.

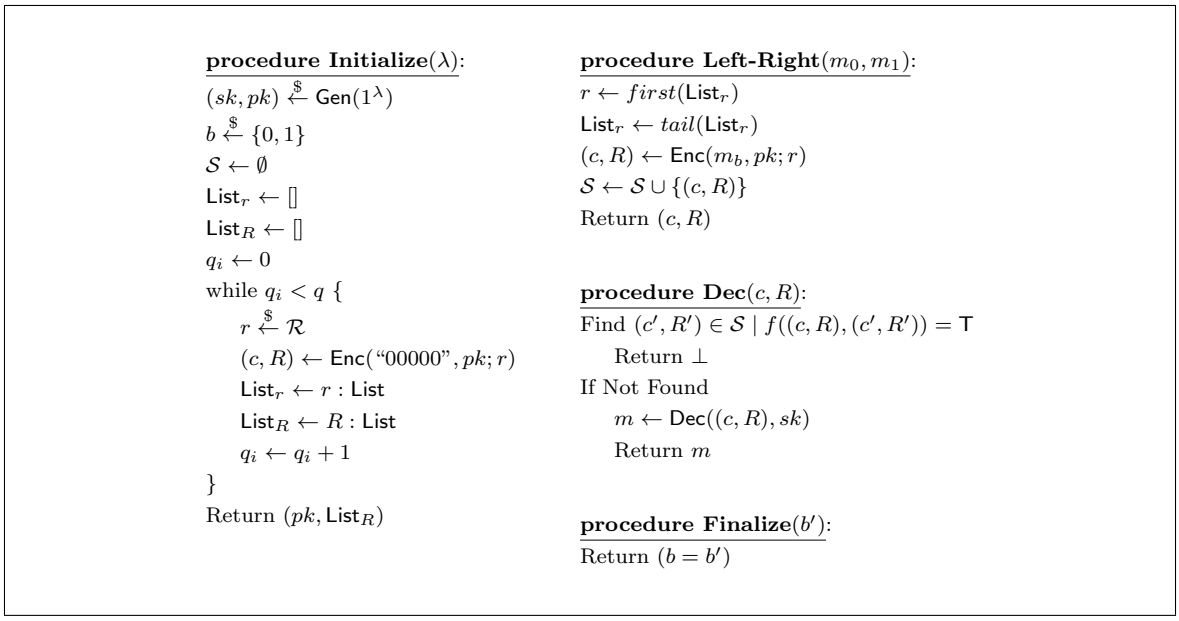

Figure 5.6: IND-RDA-q for a public-key encryption $\mathcal{E}$

Using a so-called "hybrid argument", we prove that the advantage of $\mathcal{B}$ against IND-RDA-q is at most q times the advantage of an of adversary $\mathcal{A}$ which queries the left-or-right oracle only once.

Theorem 4. $\mathbf{Adv}^{\mathsf{IND\text{-}RDA\text{-}q}}_{\mathcal{E}}(\mathcal{B}) \leq q \cdot \mathbf{Adv}^{\mathsf{IND\text{-}RDA}}_{\mathcal{E}}(\mathcal{A})$.

Proof. First, we associate $\mathcal{B}$ with the sequence of games $Game_0$, $Game_1$, ..., $Game_q$. These games only differ, relatively to game IND-RDA-q, on the behavior of the left-or-right oracle: instead of throwing a coin to decide whether to encrypt m_0 or m_1, the oracle encrypts the message on the left the first $q - i$ times the adversary queries it, and thenceforth encrypts the message on the right, being i the index of the game.

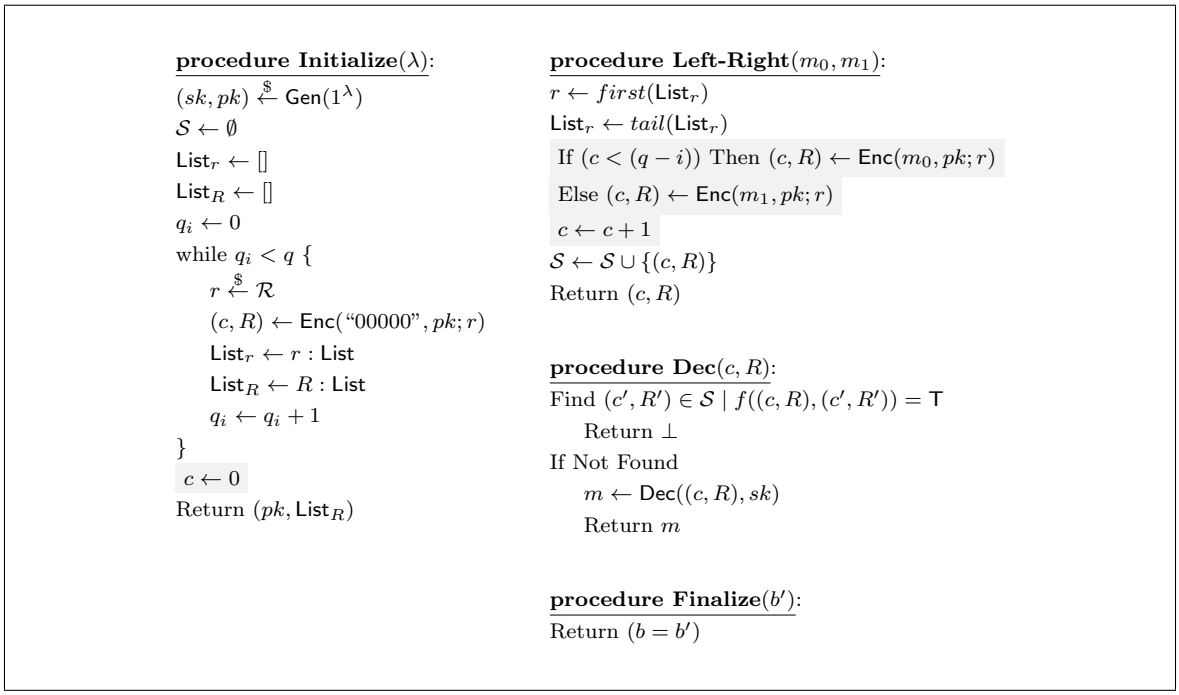

Figure 5.7: $\mathsf{Game_i}$

Let $\mathrm{P_i} = \Pr[\mathcal{B}\ outputs\ 1\ in\ \mathsf{Game_i}]$. Now observe that:

$$
\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA\text{-}q}} &\stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}RDA\text{-}q}_{\mathcal{E}}^{\mathcal{B}} \Rightarrow \mathsf{T}] - 1 \\
&= 2 \cdot \Pr[\mathcal{B}\ outputs\ b] - 1 \\
&= 2 \cdot (\Pr[\mathcal{B}\ outputs\ 0 \wedge b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \wedge b = 1]) - 1 \\
&= 2 \cdot \Pr[\mathcal{B}\ outputs\ 0 \wedge b = 0] + 2 \cdot \Pr[\mathcal{B}\ outputs\ 1 \wedge b = 1] - 1 \\
&= 2 \cdot \Pr[b = 0] \cdot \Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \\
&\quad\ 2 \cdot \Pr[b = 1] \cdot \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - 1 \\
&= \Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - 1 \\
&= \Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - \\
&\quad\ (\Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \mid b = 0]) \\
&= \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - \Pr[\mathcal{B}\ outputs\ 1 \mid b = 0] \\
&= \mathrm{P_q} - \mathrm{P_0}
\end{aligned}
$$

To continue this proof, we show that there exists a program $\mathcal{A}_i$ (Figure 5.8) such that $\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i) = \mathrm{P_{i+1}} - \mathrm{P_i}$. This program simulates the environment of $\mathsf{Game_i}$ or $\mathsf{Game_{i+1}}$ for $\mathcal{B}$. If the behavior of $\mathcal{B}$ changes significantly between the two games, it means that $\mathcal{B}$ must be gaining some advantage in game $\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}$.

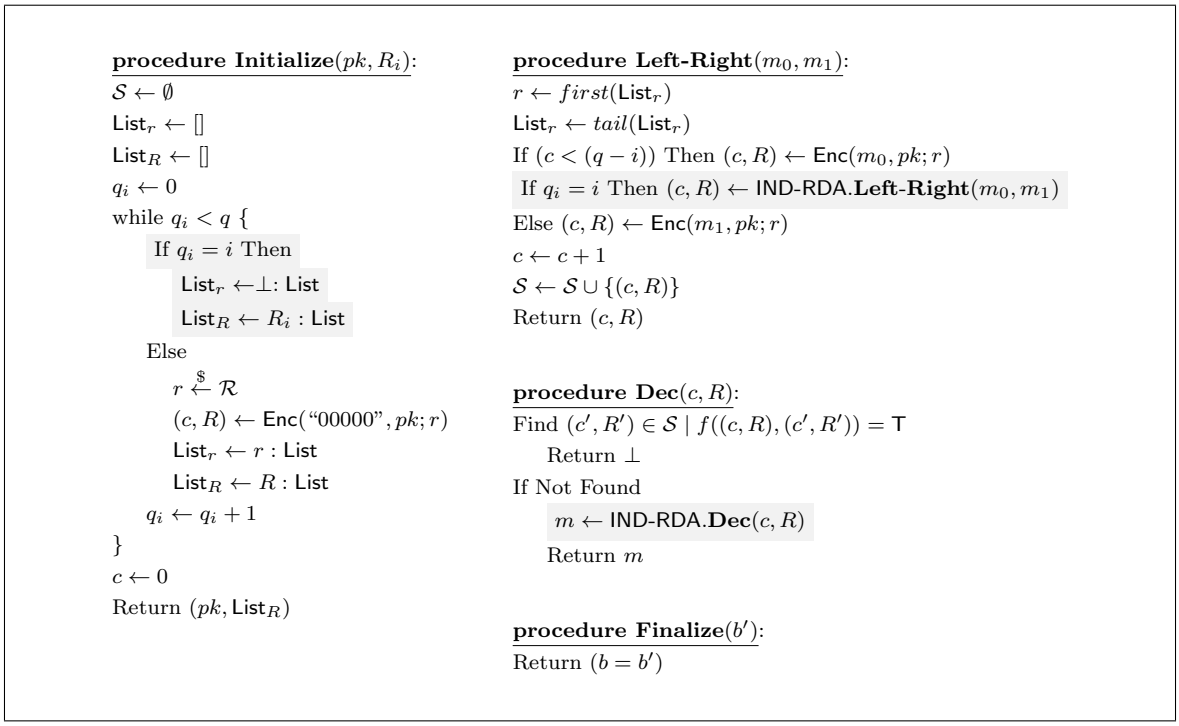

Figure 5.8: Program $\mathcal{A}_i$

From here, with a few calculations we have that

$$\begin{aligned}
& \sum_{i=0}^{q-1}(\mathrm{P_{i+1}} - \mathrm{P_i}) && = && \sum_{i=0}^{q-1}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)) \\
\Leftrightarrow \quad & \sum_{i=0}^{q-1}(\mathrm{P_{i+1}}) - \sum_{i=0}^{q-1}(\mathrm{P_i}) && = && \sum_{i=0}^{q-1}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)) \\
\Leftrightarrow \quad & \sum_{i=1}^{q}(\mathrm{P_i}) - \sum_{i=0}^{q-1}(\mathrm{P_i}) && = && \sum_{i=0}^{q-1}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)) \\
\Leftrightarrow \quad & \mathrm{P_q} - \mathrm{P_0} && = && \sum_{i=0}^{q-1}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)) \\
\Leftrightarrow \quad & \mathrm{P_q} - \mathrm{P_0} && \leq && q \cdot \max_{0 \leq i < q}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)).
\end{aligned}$$

Moreover, let

$$\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(A) = \max_{0 \leq i < q}(\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{A}_i)).$$

Finally, we have that

$$\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA\text{-}q}}(B) \leq q \cdot \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(A),$$

which concludes our proof. □

5.5 New theorems and proofs

Closely following theorems 2 and 3 of [5], we now show how these can be altered to capture the effects of randomness reuse, and achieve standard and stronger notions of security (IND-iCCA2 and sUF-iCMA).

Theorem 5. *If $\mathcal{E}$ is* IND-CPA *and reproducible, and $\mathcal{S}$ is* UF-RDA*, reproducible and satisfies Property 1, then $\mathcal{E}t\mathcal{S}$ with randomness reuse is* IND-oCCA2 *and* sUF-iCMA*.*

First, let us state what property 1 is. Informally, we say that for any given R and m, there is only one σ for which (σ, R) is a valid signature of m. This property simply says that, for each message, the set of valid signatures and the set of randomnesses are one-to-one. Again, we show later that this is a natural property in signature schemes.

Property 1. $\forall(m, \sigma, (sk, pk)), \mathsf{Sign}(m, sk) = (\sigma', R) \wedge \sigma \neq \sigma' \Rightarrow \mathsf{Verify}(m, (\sigma, R), pk) = \mathsf{F}$

The theorem then follows from Lemmas 5.1 and 5.2.

Lemma 5.1. *If $\mathcal{E}$ is* IND-CPA *and reproducible, and $\mathcal{S}$ is* UF-RDA*, reproducible and satisfies Property 1, then $\mathcal{E}t\mathcal{S}$ with randomness reuse is* IND-oCCA2*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 5.9, which is game IND-oCCA2 expanded according to the $\mathcal{E}t\mathcal{S}$ construction with randomness reuse described in Section 5.2. For simplicity of exposition, we do *not* expand the key-generation algorithm.

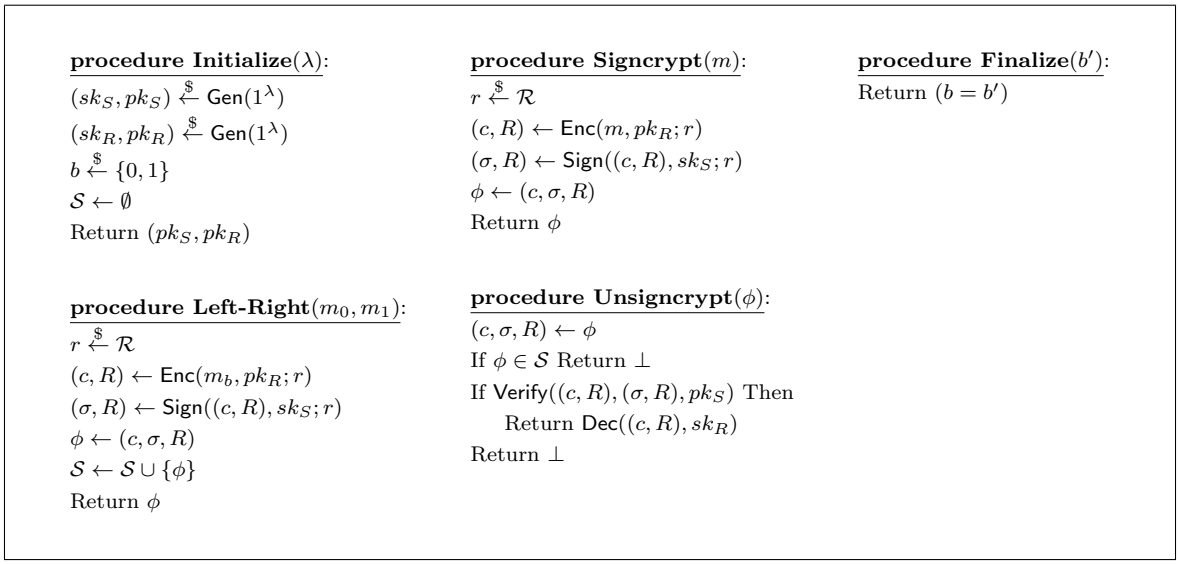

Figure 5.9: Game_0 defines IND-oCCA2 for $\mathcal{E}t\mathcal{S}$ with randomness reuse

In Game_1 (Figure 5.10) we save the replies of the signcryption oracle, and before decrypting any query of the unsigncryption oracle we check if the answer is already known. This is a bridging step from Game_0 and from the adversary's point of view, nothing really changes.

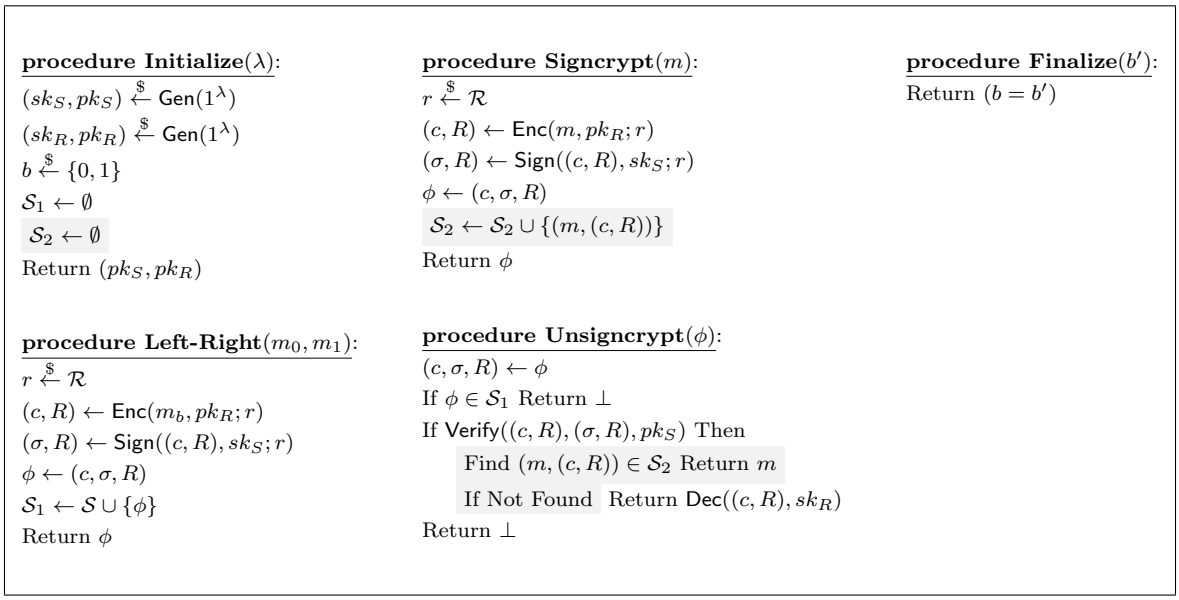

Figure 5.10: Game_1

Property 1 allows us to rephrase the environment of Game_1, and from the adversary's point of view, no part of the environment actually changes. This bridging step will lead to Game_2 (Figure 5.11).

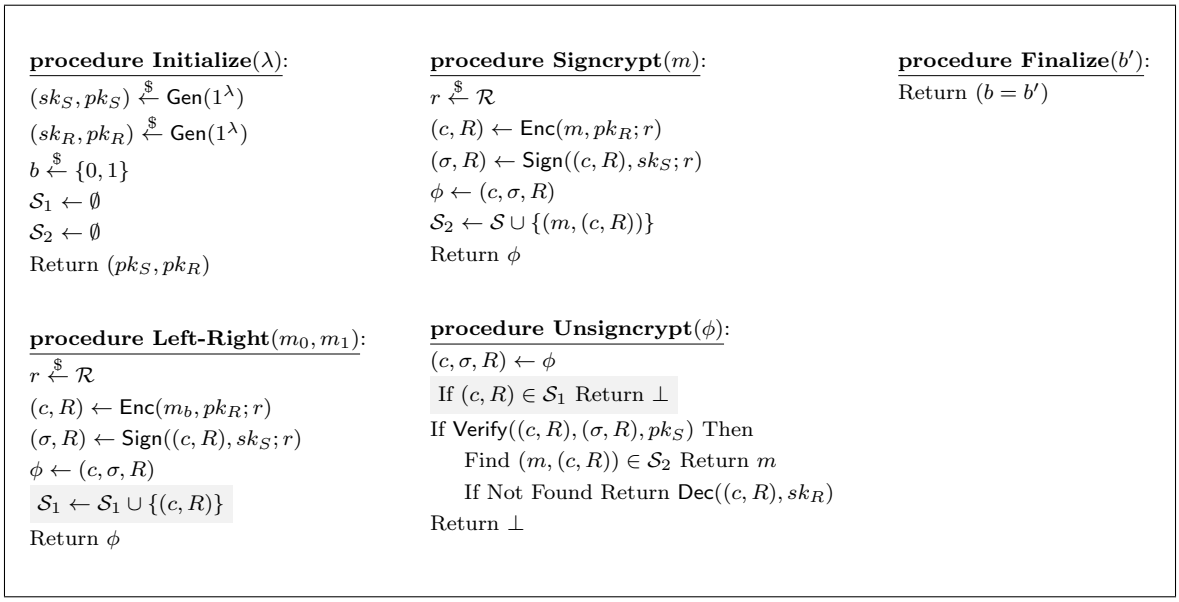

Figure 5.11: Game_2

In Game_3, the line which makes use of the decryption algorithm is removed from the simulation of the unsigncryption oracle. Let E be the event where the unsigncryption oracle uses the decryption algorithm in Game_2. The adversary $\mathcal{A}$ interacts with Game_3 (Figure 5.12) exactly the same way it interacts with Game_2 unless the event E occurs. This transition is based on the failure event E, and as a result of the difference lemma (see Section 2.2), $|\Pr[\mathsf{Game}_3 \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}]| \leq \Pr[E]$.

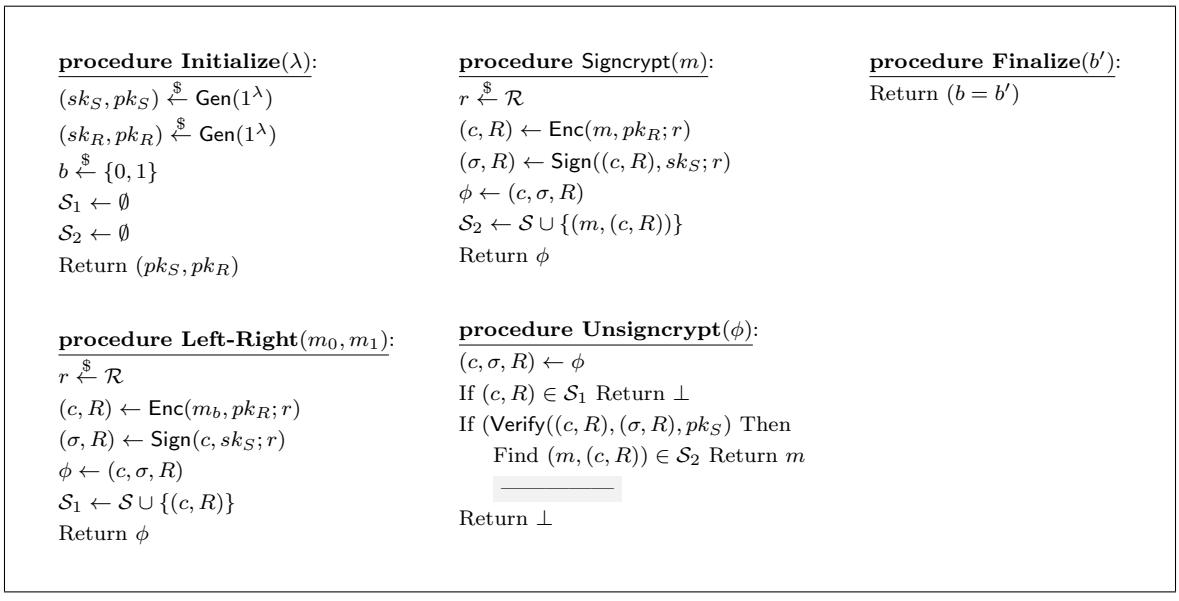

Figure 5.12: Game_3

To complete the step, we must calculate an upper-bound of $\Pr[E]$. Intuitively, in order to make the event E occur, $\mathcal{A}$ has to break the signature scheme $\mathcal{S}$. We argue this by building a program $\mathcal{B}$ (Figure 5.13) that simulates the environment of Game_2 and breaks UF-RDA (Figure 5.5) whenever event E occurs. Thus, $\Pr[E] = \mathbf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}RDA}}(\mathcal{B})$.

procedure Initialize(pk_S, List_R):
$(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{S}_1 \leftarrow \emptyset$
$\mathcal{S}_2 \leftarrow \emptyset$
Return (pk_S, pk_R)

procedure Left-Right(m_0, m_1):
$R \leftarrow first(\mathsf{List}_R)$
$\mathsf{List}_R \leftarrow tail(\mathsf{List}_R)$
$(c, R) \leftarrow \mathsf{Rep}_{\mathcal{E}}(m_b, sk_R, R)$
$(\sigma, R) \leftarrow \mathsf{UF\text{-}RDA}.\mathbf{Sign}((c, R), R)$
$\phi \leftarrow (c, \sigma, R)$
$\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{(c, R)\}$
Return ϕ

procedure Signcrypt(m):
$R \leftarrow first(\mathsf{List}_R)$
$\mathsf{List}_R \leftarrow tail(\mathsf{List}_R)$
$(c, R) \leftarrow \mathsf{Rep}_{\mathcal{E}}(m, sk_R, R)$
$(\sigma, R) \leftarrow \mathsf{UF\text{-}RDA}.\mathbf{Sign}((c, R), R)$
$\phi \leftarrow (c, \sigma, R)$
$\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{(m, (c, R))\}$
Return ϕ

procedure Unsigncrypt(ϕ):
$(c, \sigma, R) \leftarrow \phi$
If $(c, R) \in \mathcal{S}_1$ Return $\perp$
If $\mathsf{Verify}((c, R), (\sigma, R), pk_S)$ Then
  Find $(m, (c, R)) \in \mathcal{S}_2$ Return m
  If Not Found $\mathsf{UF\text{-}RDA}.\mathbf{Finalize}((c, R), (\sigma, R))$
Return $\perp$

procedure Finalize(b'):
Return

Figure 5.13: Program $\mathcal{B}$ (Lemma 5.1)

Notice that in Game_3 the unsigncryption oracle is useless to the adversary since it only returns $\perp$, or messages queried by the adversary to the signcryption oracle. With this in mind, we construct a program $\mathcal{C}$ (Figure 5.14) that perfectly simulates Game_3 and wins game IND-CPA (Figure 3.1) whenever adversary $\mathcal{A}$ wins Game_3. Thus, $\mathbf{Adv}^{\mathsf{Game}_3}(\mathcal{A}) = \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND-CPA}}(\mathcal{C})$.

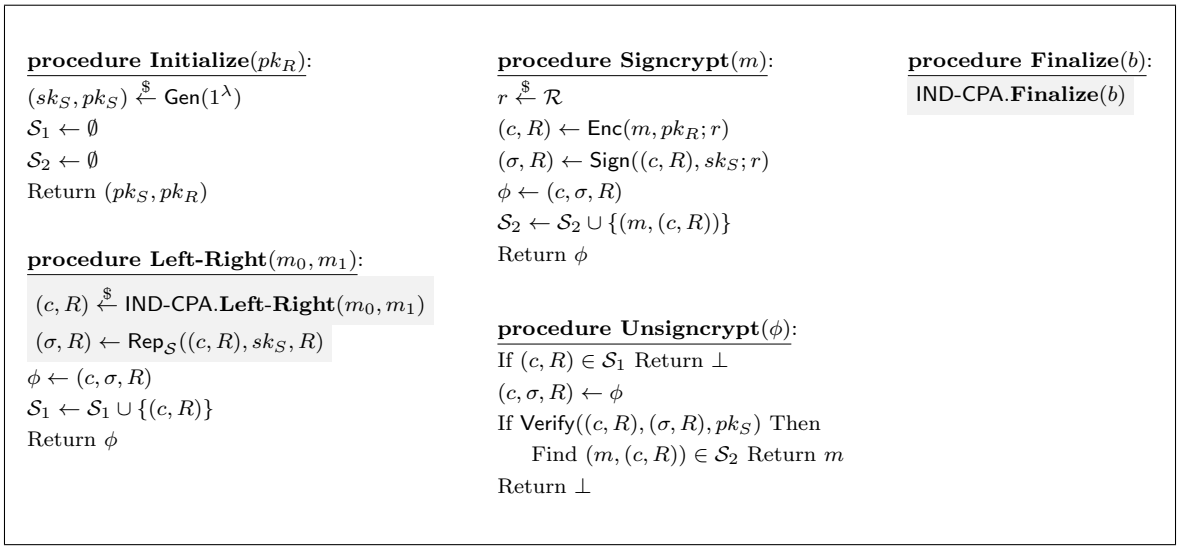

Figure 5.14: Program $\mathcal{C}$ (Lemma 5.1)

Therefore: $\mathbf{Adv}^{\mathsf{IND\text{-}ogCCA2}}_{\mathcal{E}t\mathcal{S}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}RDA}}_{\mathcal{S}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E}}(\mathcal{C})$.

□

Lemma 5.2. *If $\mathcal{E}$ is reproducible, and $\mathcal{S}$ is* UF-RDA *and satisfies Property 1, then $\mathcal{E}t\mathcal{S}$ with randomness reuse is* sUF-iCMA.

Proof. Let $\mathcal{A}$ be the adversary that plays against Game_0 defined in Figure 5.15, which is game sUF-iCMA expanded according to the $\mathcal{E}t\mathcal{S}$ construction with randomness reuse described in Section 5.2.

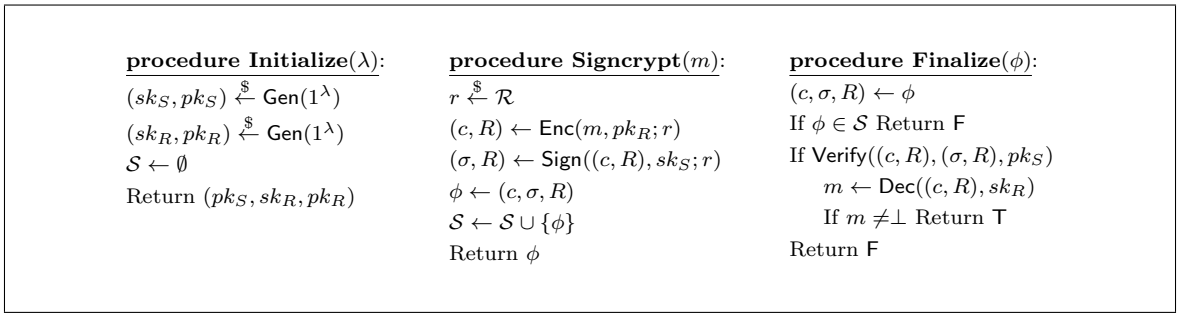

Figure 5.15: Game_0 defines sUF-iCMA for $\mathcal{E}t\mathcal{S}$ with randomness reuse

Since the signature $\mathcal{S}$ satisfies Property 1, we can rephrase the environment of Game_0, and from the adversary's point of view, no part of the environment actually changes. This bridging step will lead to Game_1 (Figure 5.16).

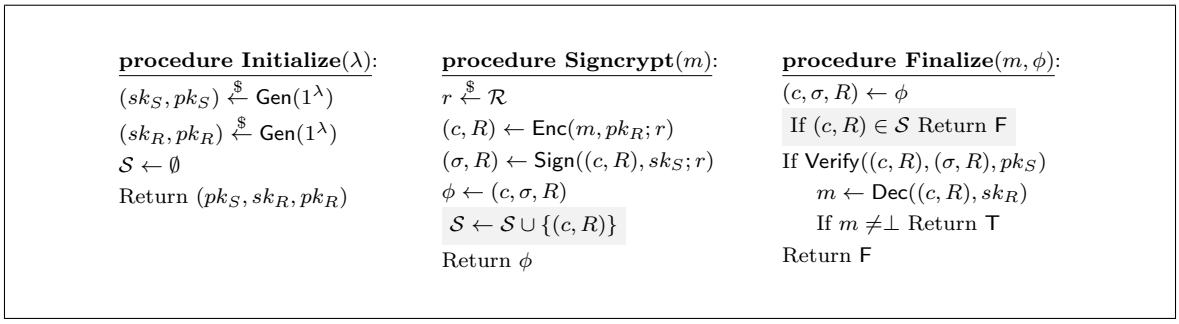

Figure 5.16: Game_1

Now, we construct a program $\mathcal{B}$ (Figure 5.17) that simulates Game_1 (Figure 5.16) and breaks UF-RDA (Figure 5.5) of the signature $\mathcal{S}$ each time $\mathcal{A}$ wins its game.

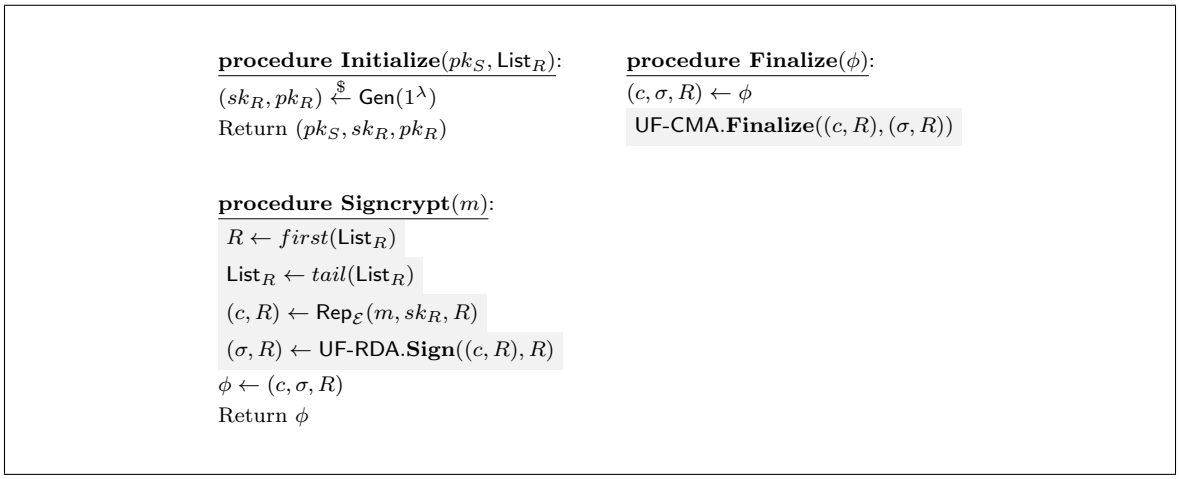

Figure 5.17: Program $\mathcal{B}$ (Lemma 5.2)

Therefore: $\mathbf{Adv}^{\mathsf{sUF\text{-}iCMA}}_{\mathcal{E}t\mathcal{S}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{UF\text{-}RDA}}_{\mathcal{S}}(\mathcal{B})$.

□

Theorem 6. *If $\mathcal{E}$ is* IND-CCA2 *and $\mathcal{S}$ is reproducible and satisfies Property 1, then $\mathcal{E}t\mathcal{S}$ with randomness reuse is* IND-iCCA2.

As opposed to what happens in game IND-oCCA2, a signcryption oracle is no longer necessary since an insider adversary knows the sender's private key. The sequential encrypt-then-sign construction presented by An et al. does not achieve IND-iCCA2 because the adversary is able produce a new signature on the challenge-ciphertext, and submit the signcrypion to the unsigncryption oracle, revealing the message underneath. This attack is not possible for signcryption schemes contructed within our framework. In fact, the only property required for the signature scheme states that for each message,

the set of valid signatures and the set of randomness are one-to-one, which mandatorily creates an extra bound between the ciphertext and the signature.

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 5.18, which is game IND-iCCA2 expanded according to the $\mathcal{E}t\mathcal{S}$ construction with randomness reuse.

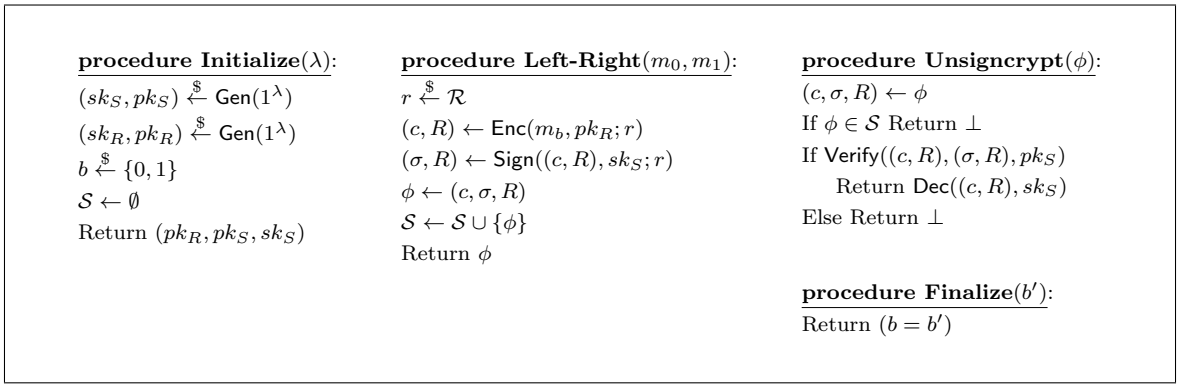

Figure 5.18: Game_0 defines IND-iCCA2 for $\mathcal{E}t\mathcal{S}$ with randomness reuse

Because the signature scheme $\mathcal{S}$ satisfies Property 1, we can rephrase the environment of Game_0. This bridging step will lead to Game_1.

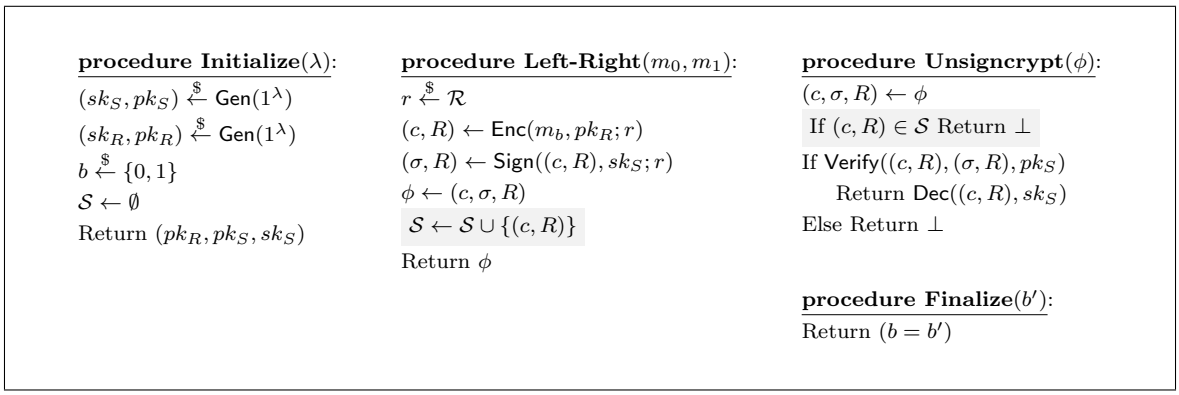

Figure 5.19: Game_1

With the help of algorithm $\mathsf{Rep}_{\mathcal{S}}$, we construct a program $\mathcal{B}$ that perfectly simulates the environment of Game_1, and wins game IND-CCA2 (Figure 3.3) every time $\mathcal{A}$ wins Game_1.

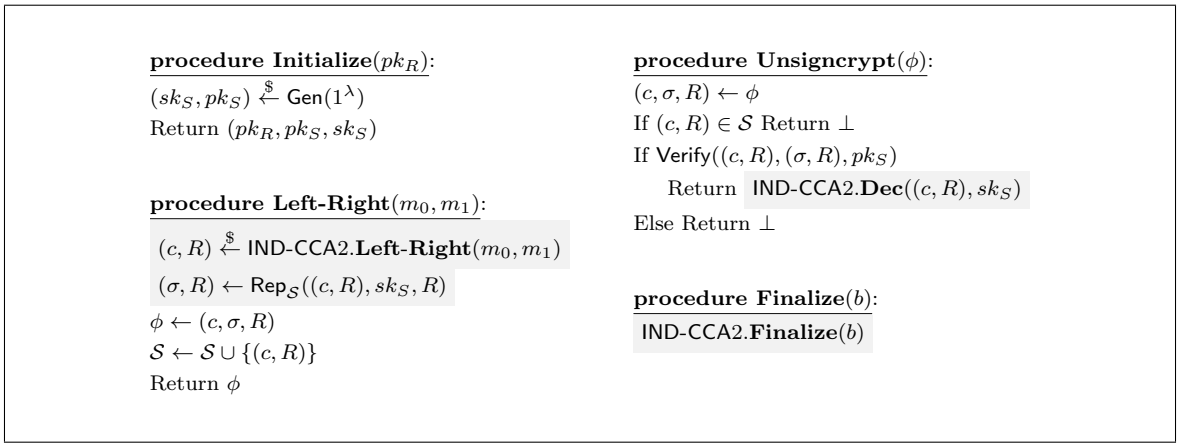

Figure 5.20: Program $\mathcal{B}$

Therefore, $\mathbf{Adv}^{\mathsf{IND\text{-}iCCA2}}_{\mathcal{E}t\mathcal{S}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{E}}(\mathcal{B})$.

□

Theorem 7. *If $\mathcal{E}$ is* IND-RDA*, reproducible and satisfies property 2, and $\mathcal{S}$ is* UF-NMA *and reproducible, then $\mathcal{S}t\mathcal{E}$ with randomness reuse is* IND-iCCA2 *and* sUF-oCMA*.*

First, let us state what Property 2 is. Informally, we say that for any given R, there is only one c that decrypts to each message m. If a reproducible scheme does not possess this property, there is an easy fix for it: Dec decrypts (c, R) as usual to obtain a message m, then it reproduces the ciphertext again with $\mathsf{Rep}(m, sk, R)$ and compares the ciphertexts, returning $\perp$ if they differ.

Property 2. $\forall(m, c, (sk, pk)), \mathsf{Enc}(m, pk) = (c', R) \wedge c \neq c' \Rightarrow \mathsf{Dec}((c, R), sk) \neq m$

The theorem then follows from Lemmas 7.1 and 7.2.

Lemma 7.1. *If $\mathcal{E}$ is* IND-RDA *and satisfies Property 2, and $\mathcal{S}$ is* UF-NMA *and reproducible, then $\mathcal{S}t\mathcal{E}$ with randomness reuse is* sUF-oCMA*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 5.21, which is identical to game sUF-oCMA expanded according to the $\mathcal{S}t\mathcal{E}$ construction with randomness reuse.

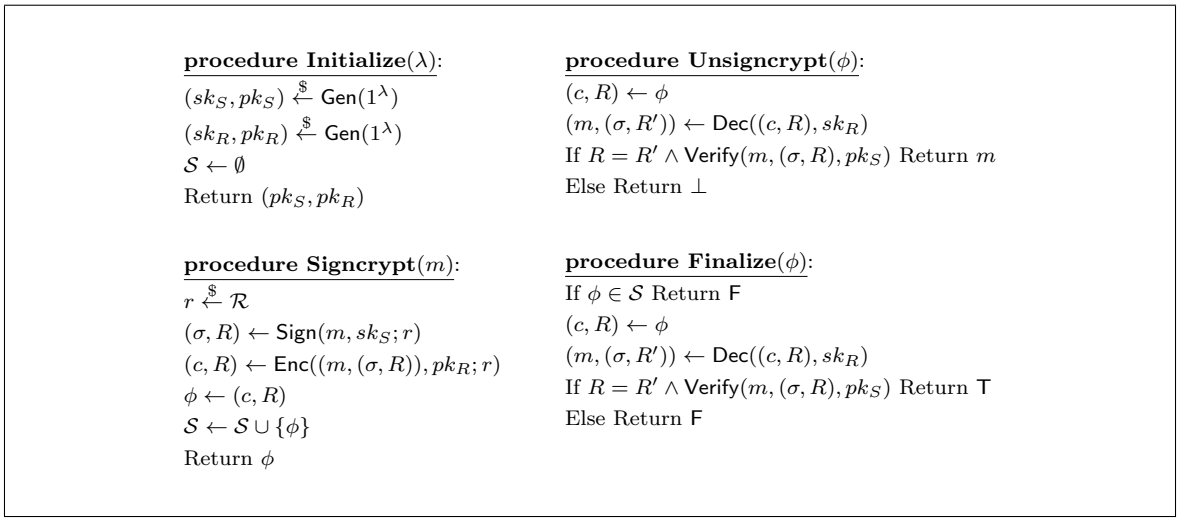

Figure 5.21: Game_0 defines sUF-oCMA for $\mathcal{S}t\mathcal{E}$ with randomness reuse

In Game_1 (Figure 5.22) we save the replies of the signcryption oracle, and before decrypting any query of the unsigncryption oracle we check if the answer is already known. This is a bridging step from Game_0 and from the adversary's point of view, nothing really changes.

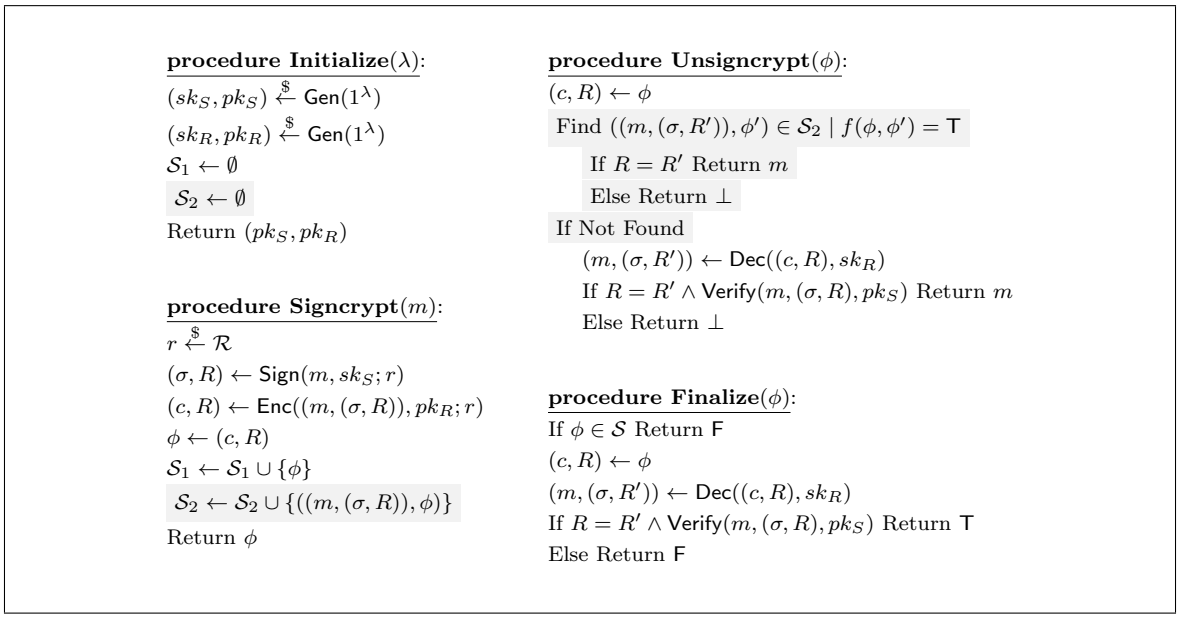

Figure 5.22: Game_1

In Game_2 (Figure 5.23), instead of correctly computing the signature of the message for the simulation of the signcryption oracle, the challenger samples a new ephemeral key and signs the message with this fake key. Although the content of the signature

becomes irrelevant, its length must be equal to that of the signature produced with the sender's private key[4]. This is a transition based on indistinguishability. Intuitively, $\mathcal{A}$ cannot detect this change (except with negligible probability) because the signature is then encrypted (together with the message) and $\mathcal{E}$ is IND-RDA.

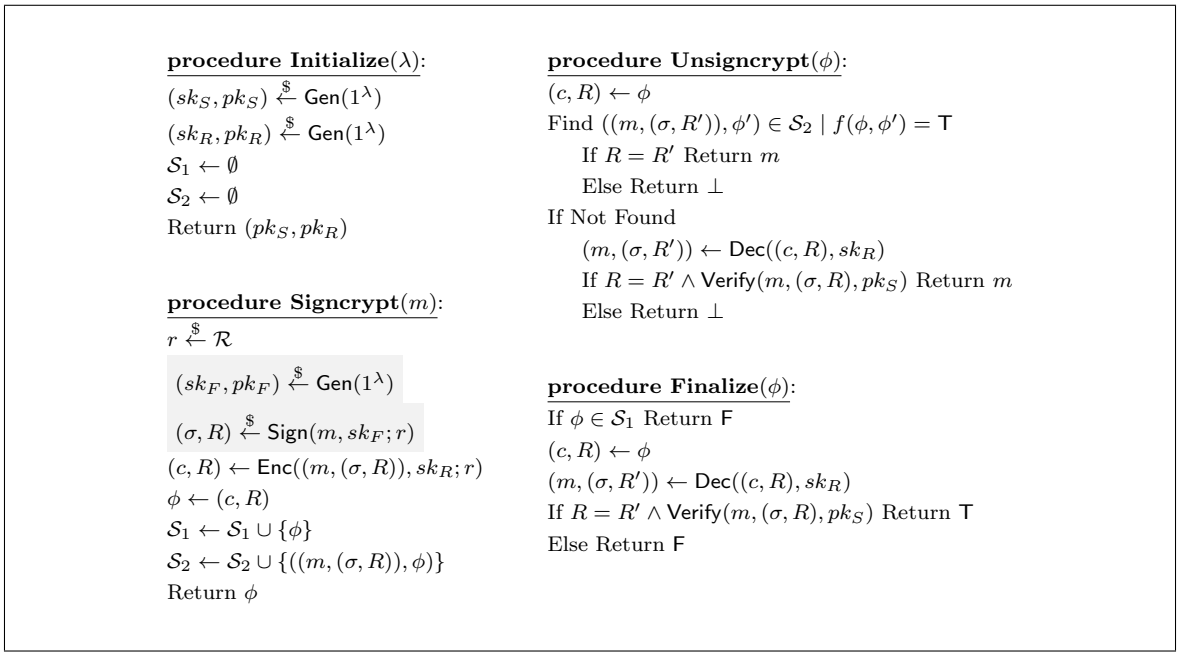

Figure 5.23: Game_2

To build a distinguishing program $\mathcal{B}$ that interpolates between Game_1 and Game_2, the game that $\mathcal{B}$ plays must allow the left-or-right oracle to be queried several times and R to be known *a priori* for the correct simulation of the signcryption oracle. As stated in the theorem, $\mathcal{E}$ is IND-RDA, but $\mathcal{B}$ must play IND-RDA-q to be able to query the left-or-right oracle more than once. However, we showed in Sub-Section 5.4.1 that there exists a program $\mathcal{C}$ for which $\mathbf{Adv}^{\mathsf{IND\text{-}RDA\text{-}q}}_{\mathcal{E}}(\mathcal{B}) \leq q \cdot \mathbf{Adv}^{\mathsf{IND\text{-}RDA}}_{\mathcal{E}}(\mathcal{C})$, being q the maximum number of times that $\mathcal{B}$ queries the left-or-right oracle. By building a distinguishing program $\mathcal{B}$ that interpolates between Game_1 and Game_2, we argue that $\Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}] = \mathbf{Adv}^{\mathsf{IND\text{-}RDA\text{-}q}}_{\mathcal{E}}(\mathcal{B}) \leq q \cdot \mathbf{Adv}^{\mathsf{IND\text{-}RDA}}_{\mathcal{E}}(\mathcal{C})$.

[4]Instead of producing a real signature over a random key, we could have sampled a random string of length equal to that of the signature produced with the legit key.

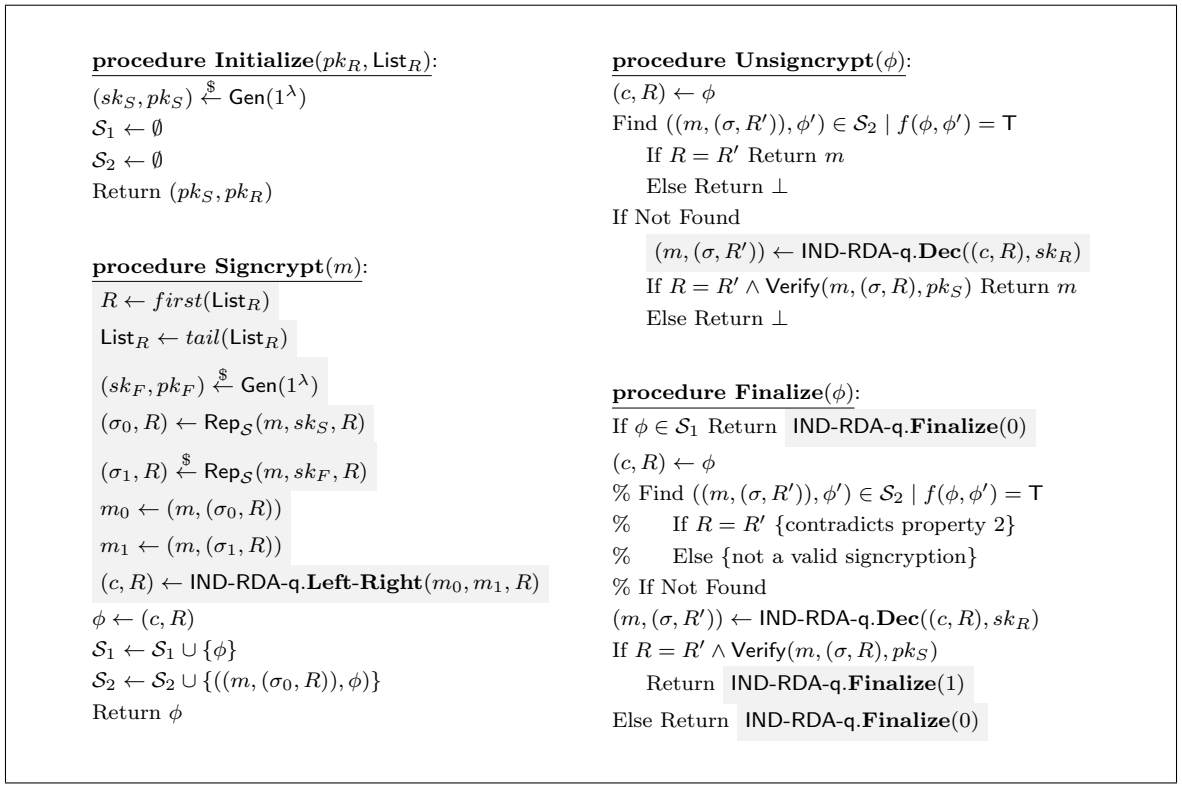

Figure 5.24: Program $\mathcal{B}$ (Lemma 7.1)

The claim follows once we observe that:

$$
\begin{aligned}
\mathbf{Adv}^{\mathsf{IND\text{–}RDA\text{–}q}}_{\mathcal{E}}(\mathcal{B}) &\stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}RDA\text{-}q} \Rightarrow \mathsf{T}] - 1 \\
&= 2 \cdot \Pr[\mathcal{B}\ outputs\ b] - 1 \\
&= 2 \cdot (\Pr[\mathcal{B}\ outputs\ 0 \wedge b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \wedge b = 1]) - 1 \\
&= 2 \cdot \Pr[\mathcal{B}\ outputs\ 0 \wedge b = 0] + 2 \cdot \Pr[\mathcal{B}\ outputs\ 1 \wedge b = 1] - 1 \\
&= 2 \cdot \Pr[b = 0] \cdot \Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \\
&\quad\ 2 \cdot \Pr[b = 1] \cdot \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - 1 \\
&= \Pr[\mathcal{B}\ outputs\ 0 \mid b = 0] + \Pr[\mathcal{B}\ outputs\ 1 \mid b = 1] - 1 \\
&= \Pr[\mathcal{A}\ outputs\ an\ invalid\ forgery\ in\ \mathsf{Game}_1] + \\
&\quad\ \Pr[\mathcal{A}\ outputs\ a\ valid\ forgery\ in\ \mathsf{Game}_2] - 1 \\
&= \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{F}] + \Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}] - 1 \\
&= (1 - \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}]) + \Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}] - 1 \\
&= \Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}]
\end{aligned}
$$

We now construct a program $\mathcal{D}$ (Figure 5.25) which simulates the environment of Game_2

for $\mathcal{A}$, and each time $\mathcal{A}$ wins Game_2, $\mathcal{D}$ wins game UF-NMA as described in Figure 3.4. Consequently, $\Pr[\mathsf{Game}_2 \Rightarrow \mathsf{T}] = \mathbf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}NMA}}(\mathcal{D})$.

procedure Initialize(λ):
$(sk_S, pk_S) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$\mathcal{S}_1 \leftarrow \emptyset$
$\mathcal{S}_2 \leftarrow \emptyset$
Return (pk_S, pk_R)

procedure Signcrypt(m):
$r \xleftarrow{\$} \mathcal{R}$
$(sk_F, pk_F) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$\sigma, R) \xleftarrow{\$} \mathsf{Sign}(m, sk_F; r)$
$(c, R) \leftarrow \mathsf{Enc}((m, (\sigma, R)), sk_R; r)$
$\phi \leftarrow (c, R)$
$\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{\phi\}$
$\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{((m, (\sigma, R)), \phi)\}$
Return ϕ

procedure Unsigncrypt(ϕ):
$(c, R) \leftarrow \phi$
Find $((m, (\sigma, R')), \phi') \in \mathcal{S}_2 \mid f(\phi, \phi') = \mathsf{T}$
  If $R = R'$ Return m
  Else Return $\perp$
If Not Found
  $(m, (\sigma, R')) \leftarrow \mathsf{Dec}((c, R), sk_R)$
  If $R = R' \wedge \mathsf{Verify}(m, (\sigma, R), pk_S)$ Return m
  Else Return $\perp$

procedure Finalize(ϕ):
If $\phi \in \mathcal{S}_1$ Return F
$(c, R) \leftarrow \phi$
$(m, (\sigma, R')) \leftarrow \mathsf{Dec}((c, R), sk_R)$
If $R = R' \wedge \mathsf{Verify}(m, (\sigma, R), pk_S)$
  UF-NMA.**Finalize**$(m, (\sigma, R'))$
Else Return F

Figure 5.25: Program $\mathcal{D}$

Therefore: $\mathbf{Adv}_{\mathcal{S}t\mathcal{E}}^{\mathsf{sUF\text{-}oCMA}}(\mathcal{A}) \leq q \cdot \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}RDA}}(\mathcal{C}) + \mathbf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}NMA}}(\mathcal{D})$.

□

Lemma 7.2. *If $\mathcal{E}$ is* IND-RDA *and satisfies Property 2, and $\mathcal{S}$ is reproducible, then $\mathcal{S}t\mathcal{E}$ with randomness reuse is* IND-iCCA2*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 5.26, which is identical to game IND-iCCA2 expanded according to the $\mathcal{S}t\mathcal{E}$ construction with randomness reuse.

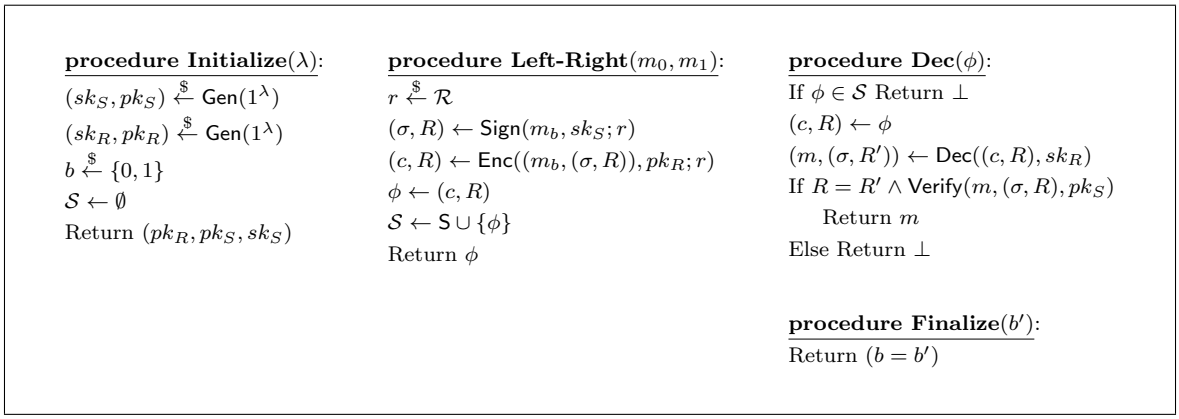

Figure 5.26: Game_0 defines IND-iCCA2 for $\mathcal{S}t\mathcal{E}$ with randomness reuse

We prove this lemma by building a program $\mathcal{B}$ (Figure 5.27) that simulates the environment of Game_0 for $\mathcal{A}$ in a way that $\mathcal{B}$ wins IND-RDA (Figure 5.4) whenever $\mathcal{A}$ wins Game_0.

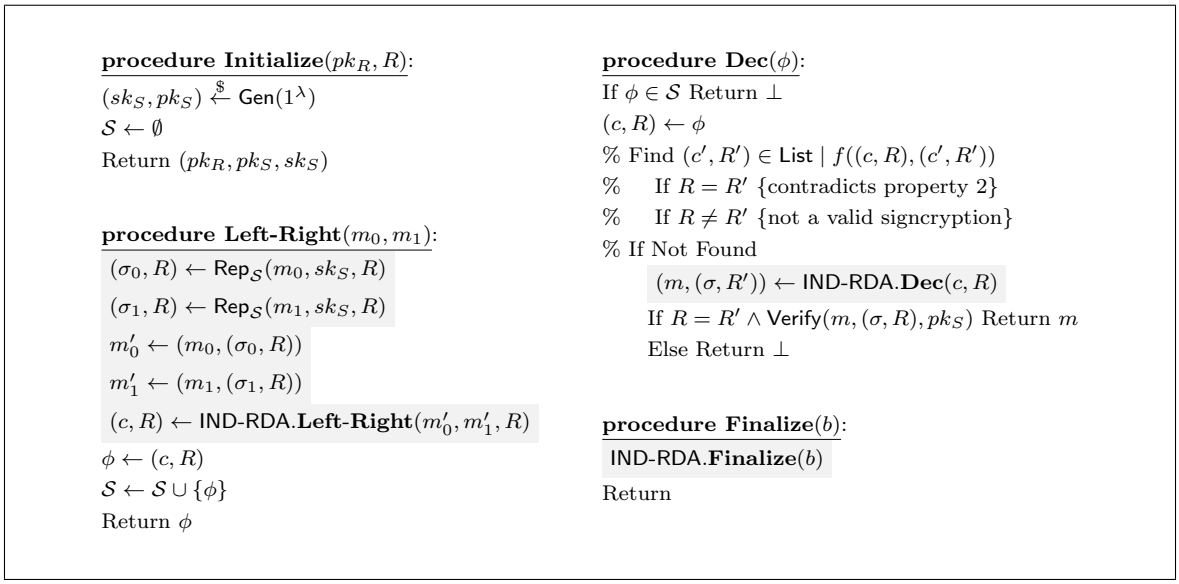

Figure 5.27: Program $\mathcal{B}$ (Lemma 7.2)

Thus, $\mathbf{Adv}^{\mathsf{IND\text{-}iCCA2}}_{\mathcal{S}t\mathcal{E}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{IND\text{-}RDA}}_{\mathcal{E}}(\mathcal{B})$.

□

Theorem 8. *If $\mathcal{E}$ is reproducible and satisfies Property 2, and $\mathcal{S}$ is* sUF-CMA*, then $\mathcal{S}t\mathcal{E}$ with randomness reuse is* sUF-iCMA*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 5.28, which is game sUF-iCMA expanded according to the $\mathcal{S}t\mathcal{E}$ construction

with randomness reuse.

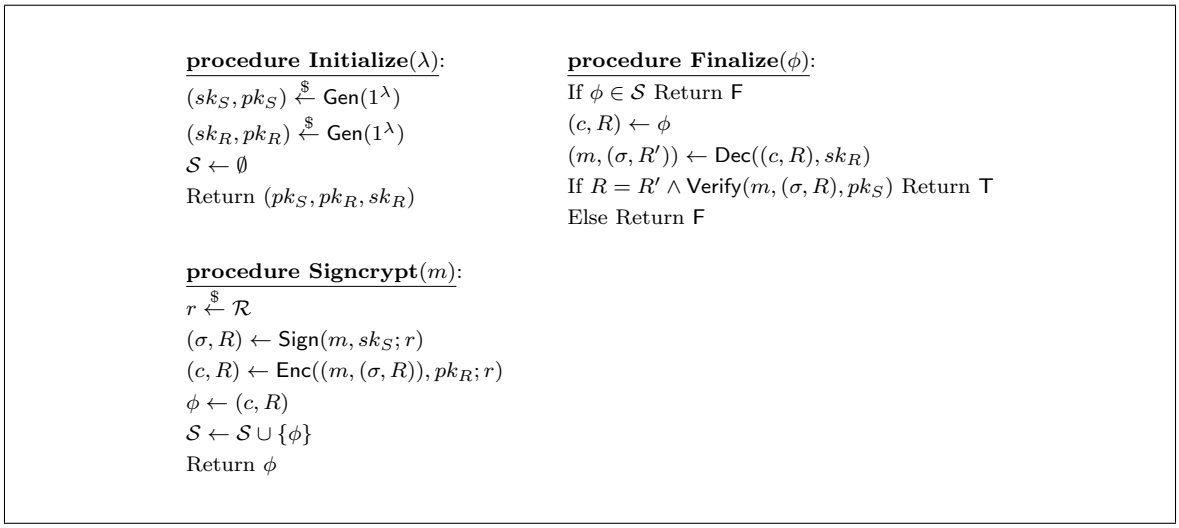

Figure 5.28: Game_0 defines sUF-iCMA for $\mathcal{S}t\mathcal{E}$ with randomness reuse

To prove this theorem, we do a simple reduction and construct a program $\mathcal{B}$ (Figure 5.29) that plays game sUF-CMA (Figure 3.6) and simulates the environment of Game_0. Each time $\mathcal{A}$ produces a valid forgery for Game_0, $\mathcal{B}$ wins game sUF-CMA. Indeed, if $\mathcal{A}$ outputs a valid forgery (c, R) it means that a) it is not in the set of signcryptions produced by the signcryption oracle of Game_0, and b) it decrypts to $(m, (\sigma, R))$ and (σ, R) is a valid signature on m. By Property 2, σ is unique, so $(m, (\sigma, R))$ is not on the set of signatures produced by the signing oracle of game sUF-CMA either.

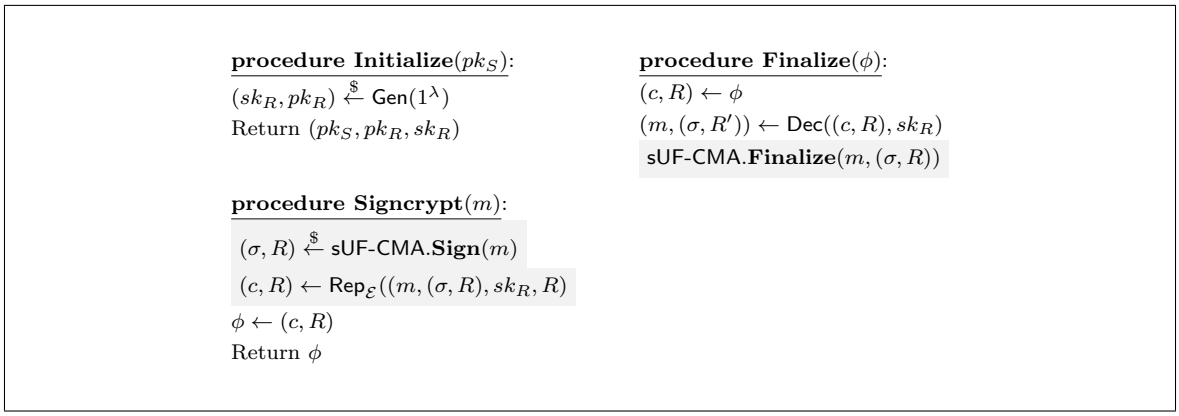

Figure 5.29: Program $\mathcal{B}$

Therefore, we have that $\mathbf{Adv}^{\mathsf{sUF\text{-}iCMA}}_{\mathcal{S}t\mathcal{E}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{sUF\text{-}CMA}}_{\mathcal{S}}(\mathcal{B})$.

□

5.6 Results summary

Tables 5.1 and 5.2 summarise our results for the $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ constructions with randomness reuse. Both full outsider security and full insider security can now be achieved if encryption and signature meet the requirements. As discussed in Section 5.2, with randomness reuse, $\mathcal{E}t\mathcal{S}$ would be preferable over $\mathcal{S}t\mathcal{E}$ because it produces shorter signcryptions.

Table 5.1: Results obtained for the $\mathcal{E}t\mathcal{S}$ construction with randomness reuse.

	IND-CPA + $\text{Rep}_{\mathcal{E}}$		IND-CCA2 + $\text{Rep}_{\mathcal{E}}$	
Property 1 + $\text{Rep}_{\mathcal{S}}$	...	...	...	...
	...	...	...	IND-iCCA2
UF-RDA + Property 1 + $\text{Rep}_{\mathcal{S}}$	sUF-oCMA	IND-oCCA2	...	...
	sUF-iCMA	...	sUF-iCMA	IND-iCCA2

Table 5.2: Results obtained for the $\mathcal{S}t\mathcal{E}$ construction with randomness reuse.

	Property 2 + $\text{Rep}_{\mathcal{E}}$		IND-RDA + Property 2 + $\text{Rep}_{\mathcal{E}}$	
UF-NMA + $\text{Rep}_{\mathcal{S}}$	...	...	sUF-oCMA	IND-oCCA2
	...	...	...	IND-iCCA2
sUF-CMA + $\text{Rep}_{\mathcal{S}}$	...	...	...	...
	sUF-iCMA	...	sUF-iCMA	IND-iCCA2

5.7 Candidates for instantiation

As stated in Section 2.4 all proofs in this dissertation are in the standard model. If we want the proposed constructions to be secure in the standard model, our framework must be instantiated with primitives proven secure in the standard model as well. First let us turn our attention to the security models in which encryption and signature must be secure. New security models for authenticity and confidentiality have been presented in this chapter, therefore, one should not expect to find in the literature schemes that have been proven to comply with the new constrains. IND-RDA is a stronger security notion than IND-gCCA2, which means that if an encryption scheme is secure in the IND-RDA model, it is also secure in IND-gCCA2 model. Still, IND-gCCA2 is not a very standard security model (and this is exactly why we try to avoid the sequential construction without randomness reuse which can only be proven secure in this weaker and non-standard model). However, we point out that encryption schemes most recently designed are IND-CCA2, which has become the standard notion of security for encryption, and IND-CCA2 is just a special case of IND-gCCA2-security, where

the decryption-respecting relation f is simply the identity function. With this in mind, schemes like Cramer-Shoup[14] and Kurosawa-Desmedt[30] appear at the top of the list as candidates which one should try to prove secure in the stronger IND-RDA model. On the signature side, and since UF-RDA is a stronger notion of security than UF-CMA, schemes secure in the latter model are the obvious candidates to try to prove secure in the UF-RDA model.

Before suggesting any concrete schemes as candidates for the signature primitive, we should look at reproducibility, which is required for both primitives in both $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ construction with randomness reuse. Many encryption schemes, including the two already referred in this section, are reproducible, and reproduction algorithms are fairly trivial[8]. Basically, it is so because reproduction algorithms for encryption take the secret key as an additional parameter beyond those taken by the respective encryption algorithms. This compensates the fact that only R is known instead of r. Taking ElGamal[21] as an example[5], the ciphertext $c = m \cdot g^{r \cdot x}$ is easily computable from $(g, m, r, pk = g^x)$ as well as from $(g, m, R = g^r, sk = x)$. Reproduction algorithms for signature take the same parameters as signature algorithms, and having the secret key is no longer a plus. So, it seems that only signature schemes for which $R = r$ are reproducible, or the schemes are not as efficient as they could be. Luckily, Boneh-Boyen signature [12] outputs r as part of the signature. Put differently, Boneh-Boyen signature is partitioned and $R = r$, which makes the scheme reproducible.

Properties 1 and 2 are mathematical properties that must hold for signature and encryption, respectively. It is not the goal of this dissertation to present concrete signcryption schemes, but rather introduce new constructions based on randomness reuse. Nevertheless, with a few calculations it is easy to verify that Property 1 holds for Boneh-Boeyn signature scheme, and that Property 2 holds for Cramer-Shoup and Kurosawa-Desmedt encryption schemes.

Finally, signature and encryption must be compatible, i.e., on input the same random coins r, the signature and encryption algorithms produce the same R. Obviously, the encryption algorithm cannot simply output r, otherwise the scheme would become trivially breakable. However, Boneh-Boyen signature algorithm can be modified in order to become compatible with Kurosawa-Desmedt encryption scheme by simply computing $r' = hash(g^r)$ and use r' the same way as r in the original scheme. The efficiency of the construction decreases with respect to the computation of an extra hash function, but this is largely compensate by the reuse of random coins and extra security gained.

[5] ElGamal encryption scheme is not IND-CCA2 but it is presented here instead of Kurosawa-Desmedt for simplicity of exposition.

Chapter 6

KEM-DEM with Randomness Reuse

In this chapter we extend the construction proposed by Cramer and Shoup to construct hybrid schemes which allow for randomness reuse between KEM and DEM components. Most symmetric-key encryption schemes are designed to be secure against multiple plaintext attacks, which clearly requires the usage of some input randomness. In block ciphers, this randomness usually takes the form of a initialization vector. The advantage of our approach in comparison to common practice is that the initialization vector no longer needs to be fixed or derived from the secret key, which permits aligning the practical usage of the KEM-DEM paradigm with existing theoretical results regarding the security of symmetric-key encryption schemes. We demonstrate the practicality of our construction with a simple instantiation of our framework.

6.1 The construction

The construction is similar to the one described in Section 3.5.3 except that now, KEM and DEM must share the same randomness space $\mathcal{R}$ from where random coins are sampled only once for both KEM and DEM components. This hybrid construction with randomness reuse is specified by the following three algorithms $\mathcal{H} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, polynomial-time-bounded in the length of their input.

- $\mathsf{Gen}(1^\lambda)$ is the same as $\mathcal{K}.\mathsf{Gen}(1^\lambda)$.
- $\mathsf{Enc}(m, pk)$ computes the sequence: $r \xleftarrow{\$} \mathcal{R}$; $(c_1, k) \leftarrow \mathcal{K}.\mathsf{Encap}(pk; r)$; $c_2 \leftarrow \mathcal{D}.\mathsf{Encap}(m, k; r)$; $c \leftarrow (c_1, c_2)$; return c.

- $\mathsf{Dec}(c, sk)$ computes the sequence: $(c_1, c_2) \leftarrow c$; $k \leftarrow \mathcal{K}.\mathsf{Decap}(c_1, sk)$; $m \leftarrow \mathcal{D}.\mathsf{Decap}(c_2, k)$; return m.

We do not require the decryption algorithm to verify that the ciphertext was constructed by instantiating KEM and DEM with the same random coins, as we did in Section 5.2 for signcryption constructions with randomness reuse. This is because we are not looking for security gains, since the construction proposed by Cramer and Shoup already achieves IND-CCA2-security. However, performance gains in terms of computational costs and bandwidth are still on the table. We study how those can be achieved without compromising the security of the construction.

6.2 Reproducibility

To pursue safely with the optimization of reusing random coins in the construction of a hybrid encryption scheme, reproducibility between KEM and DEM in both senses is required. We say that KEM is *reproducible* from DEM if there is a polynomial-time *reproduction* algorithm $\mathsf{Rep}_{\mathcal{D}\to\mathcal{K}}$, and we say that DEM is *reproducible* from KEM if there is a polynomial-time *reproduction* algorithm $\mathsf{Rep}_{\mathcal{K}\to\mathcal{D}}$, as follows:

- $\mathsf{Rep}_{\mathcal{D}\to\mathcal{K}}(sk, pk, c_2)$ is a deterministic reproduction algorithm, which on input a key air (sk, pk), and a ciphertext c_2 resulting from some call of $\mathcal{D}.\mathsf{Encap}(\,\cdot\,; r)$, outputs a pair containing a shared key $k \in \mathcal{SK}$, and a ciphertext c_1 of k encrypted under pk and the same random coins r.

- $\mathsf{Rep}_{\mathcal{K}\to\mathcal{D}}(m, k, sk, pk, c_1)$ is a deterministic reproduction algorithm, which on input a message m, a shared key $k \in \mathcal{SK}$, a key pair (sk, pk), and a ciphertext c_1 resulting from $\mathcal{K}.\mathsf{Encap}(pk; r)$, outputs a ciphertext c_2 of m encrypted under the shared key k and the same random coins r.

We show in Section 6.4 that it is possible to construct KEM and DEM such that reproduction algorithms exist in both senses, and we discuss the value of hybrid constructions with randomness reuse.

6.3 New theorem and proof

Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure 6.1, which is game IND-CCA2 expanded according to the KEM-DEM construction with randomness reuse described in Section 6.1.

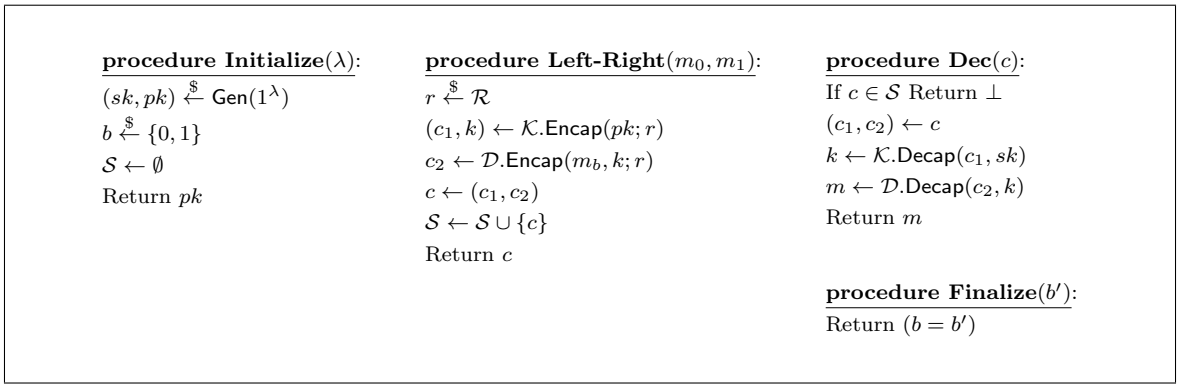

Figure 6.1: Game_0 defines IND-CCA2 for $\mathcal{K}$-$\mathcal{D}$ with randomness reuse

In Game_1, $\mathcal{D}$.Encap uses a random key to encrypt data instead of the key returned by $\mathcal{K}$.Encap. Because KEM is IND-CCA2, the adversary cannot detect this change (except with negligible probability). This transition is based on indistinguishability.

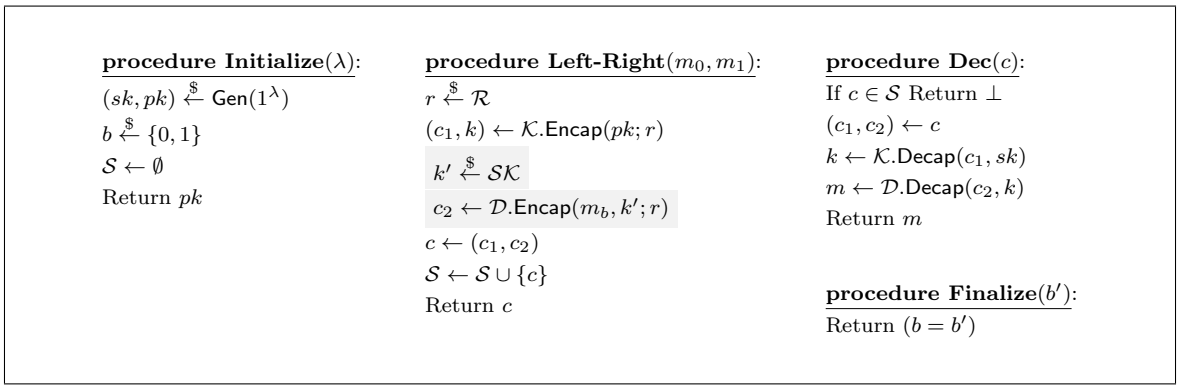

Figure 6.2: Game_1

We now build a distinguishing program $\mathcal{B}$ that interpolates between Game_0 and Game_1, and argue that $\Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_0 \Rightarrow \mathsf{T}] = \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{K}}(\mathcal{B})$. Intuitively, if the behavior of $\mathcal{A}$ changes significantly between Game_0 and Game_1, $\mathcal{B}$ breaks the IND-CCA2-security of KEM.

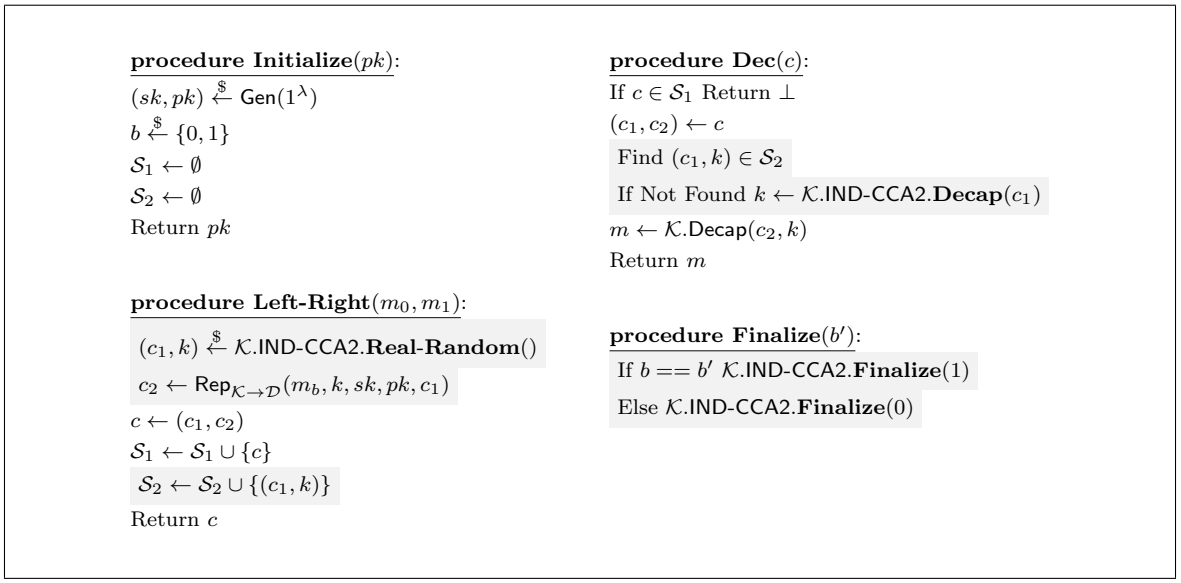

Figure 6.3: Program $\mathcal{B}$

The claim follows once we observe that:

$$
\begin{aligned}
\mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{K}}(\mathcal{B}) &\stackrel{\text{def}}{=} 2 \cdot \Pr[\mathcal{K}.\mathsf{IND\text{-}CCA2} \Rightarrow \mathsf{T}] - 1 \\
&= 2 \cdot \Pr[\mathcal{B} \textit{ outputs } b^\star] - 1 \\
&= 2 \cdot (\Pr[\mathcal{B} \textit{ outputs } 0 \wedge b^\star = 0] + \Pr[\mathcal{B} \textit{ outputs } 1 \wedge b^\star = 1]) - 1 \\
&= 2 \cdot \Pr[\mathcal{B} \textit{ outputs } 0 \wedge b^\star = 0] + 2 \cdot \Pr[\mathcal{B} \textit{ outputs } 1 \wedge b^\star = 1] - 1 \\
&= 2 \cdot \Pr[b^\star = 0] \cdot \Pr[\mathcal{B} \textit{ outputs } 0 \mid b^\star = 0] + \\
&\quad 2 \cdot \Pr[b^\star = 1] \cdot \Pr[\mathcal{B} \textit{ outputs } 1 \mid b^\star = 1] - 1 \\
&= \Pr[\mathcal{B} \textit{ outputs } 0 \mid b^\star = 0] + \Pr[\mathcal{B} \textit{ outputs } 1 \mid b^\star = 1] - 1 \\
&= \Pr[\mathcal{A} \textit{ outputs } b' \neq b \textit{ in } \mathsf{Game}_0] + \Pr[\mathcal{A} \textit{ outputs } b' = b \textit{ in } \mathsf{Game}_1] - 1 \\
&= \Pr[\mathsf{Game}_0 \Rightarrow \mathsf{F}] + \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}] - 1 \\
&= (1 - \Pr[\mathsf{Game}_0 \Rightarrow \mathsf{T}]) + \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}] - 1 \\
&= \Pr[\mathsf{Game}_1 \Rightarrow \mathsf{T}] - \Pr[\mathsf{Game}_0 \Rightarrow \mathsf{T}]
\end{aligned}
$$

Game_1 is then reduced to game IND-CCA2 of DEM. With the help of algorithm $\mathsf{Rep}_{\mathcal{D}\to\mathcal{K}}$, we construct a program $\mathcal{C}$ that perfectly simulates the environment of Game_1 and breaks IND-CCA2-security of DEM (Figure 3.12) every time $\mathcal{A}$ wins Game_1.

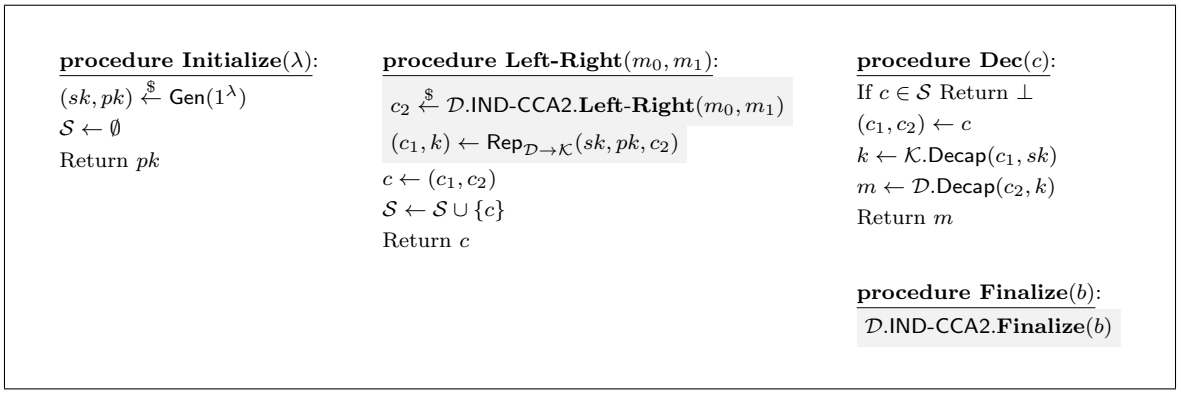

Figure 6.4: Program $\mathcal{C}$

Therefore we conclude that $\mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{K}\text{-}\mathcal{D}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{K}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{D}}(\mathcal{C})$.

6.4 Possible instantiations

Although the single plaintext attack model considered in Section 3.5.2 is sufficient for constructing secure data encapsulation mechanisms for the KEM-DEM framework developed by Cramer and Shoup, most of symmetric encryption schemes are designed to be secure against multiple plaintext attacks, where an adversary is allowed to adaptively obtain many encryptions of its choice, and not just the single ciphertext returned by the real-or-random oracle. To formalize a game in a multiple plaintext attack model, a new encryption oracle must be made available to the adversary, and the encryption algorithm can no longer be deterministic, otherwise the scheme ceases to be secure in this new security model.

The common practice is to set the initialization vector of the symmetric-key encryption and decryptions algorithms to some fixed value, which still makes the scheme secure against single plaintext attacks, and use resulting one-time symmetric-key encryption scheme as DEM.

The construction we propose allow the symmetric-key encryption scheme to remain secure against multiple plaintext attacks without a bandwidth overhead. In fact, we suggest as a possible instantiation of our framework to use Cramer-Shoup KEM [2] and AES cipher [3] in CBC mode [4] with PKCS #5 padding [39] as DEM, and set the initialization vector to $iv = hash(g^r)$, where g is a parameter of KEM and r is the randomness used to encrypt the symmetric-key k.

Since we now have a DEM which is secure against multiple plaintext attacks, whether

it is possible or not to further optimize the hybrid encryption scheme described above is a question that we leave open.

Chapter 7

Conclusions and Future Directions

The aim of this dissertation was to extend the existing results on randomness reuse and study its application across different primitives. More specifically, we sought to show under which conditions can randomness be safely shared between signature and encryption, and between key encapsulation mechanisms and data encapsulation mechanisms.

In Chapter 5 we introduced two new security models: IND-RDA for encryption and UF-RDA for signature. These models were built on top of IND-gCCA2 and UF-CMA, with the particularity that the attack models have been strengthened in order to allow the adversary to adaptively query its oracles based on some information regarding the randomness state. The idea was to closely follow the results from An et al. [5] but, in our opinion, IND-gCCA2 is a definition that should be avoided due to the complexity of the definition itself and because it is a non-standard definition. However, if we consider the case where the decryption-respecting relation f is the identity function, the model is identical to IND-CCA2, and our IND-RDA security model becomes much simpler too. We suggested to instantiate our construction with Kurosawa-Desmedt encryption scheme, which is the most efficient encryption scheme to date proven secure in the standard model, and with a slightly modified version of Boneh-Boeyn signature scheme, which is one of the most efficient signature schemes to date to have been proven secure in the standard model. We haven't proved that neither Kurosawa-Desmedt encryption scheme is IND-RDA or Boneh-Boeyn signature scheme if UF-RDA. This is a question that still needs to be investigated. If encryption is IND-RDA then, using the sign-then-encrypt construction with randomness reuse, we could construct a signcryption scheme fully secure against insider attackers. If the signature is UF-RDA then the encrypt-then-sign construction with randomness reuse is the way to go. If both encryption and signature

are secure in these new models, then it would be preferable to encrypt-then-sign because the signcryptions are shorter by the length of R, and the reduction of the proof is tighter. In fact, notice that encrypting then signing with partitioned schemes produces signcryptions of the form $((c, R), (\sigma, R'))$. But, because we require the schemes to be compatible in our construction, it must be that $R = R'$ and we don't need to transmit R twice.

Although any instantiation of our constructions will most probably end up with an extra hash execution when compared to plain sequential composition of signature and encryption, this slight damage to performance is largely compensated by the reuse of randomness. More importantly, our framework serve as base for the construction of signcryption schemes which also benefit from security enhancements. In fact, signcryption schemes fully secure against insider attackers in the standard model and according to the strongest common definitions of security for confidentiality and authenticity, which are IND-CCA2 and sUF-CMA, result from the instantiation of our framework.

Since no efficient and deterministic signature scheme is known to be secure in the standard model [27], which would potentially lead a plain sequential encrypt-then-sign construction full insider secure in the standard model, our framework is probably the first step towards the construction of such signcryption schemes.

Although beyond of the scope of this dissertation, we point out that the same extra bound achieved by randomness reuse in the proposed constructions is attainable with a sequential sign-then-encrypt construction if encryption is IND-RDA and the signature algorithm signs R together with the message m, where R is the state information regarding the random coins to be used by the encryption algorithm.

The construction we proposed for KEM-DEM allows the symmetric-key encryption scheme to remain secure against multiple plaintext attacks without a bandwidth overhead, and the common practice of fixing the value of the initialization vector is no longer needed. This however costs an extra hash function to be computed, and it remains unclear how could we take advantage of having a DEM which is secure against multiple plaintext attacks.

Overall, we demonstrated through rigorous code-based game-hopping proofs that randomness can be reused across different primitives in specific circumstances, and that not only this may provide efficiency gains, but may also lead to additional security guarantees.

Appendix A

Proofs of Theorems 2 and 3

A.1 Proof of theorem 2

The theorem follows from lemmas 2.1 and 2.2.

Lemma 2.1. *If* $\mathcal{E}$ *is* IND-CPA *and* $\mathcal{S}$ *is* UF-CMA*, then* $\mathcal{E}t\mathcal{S}$ *is* IND-ogCCA2[1].

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time (outsider) adversary against Game_0 defined in Figure A.1, which is game IND-ogCCA2 expanded according to the $\mathcal{E}t\mathcal{S}$ construction, where $f((c_1, \sigma_1), (c_2, \sigma_2)) = \mathsf{T} \Leftrightarrow c_1 = c_2$. Let $\Pr[S_i]$ be the probability Game_i returns T.

procedure Initialize(λ):
$(sk_S, pk_S) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$b \xleftarrow{\$} \{0, 1\}$
$\mathcal{S}_1 \leftarrow \emptyset$
$\mathcal{S}_2 \leftarrow \emptyset$
Return (pk_S, pk_R)

procedure Left-Right(m_0, m_1):
$c \xleftarrow{\$} \mathsf{Enc}(m_b, pk_R)$
$\sigma \xleftarrow{\$} \mathsf{Sign}(c, sk_S)$
$\phi \leftarrow (c, \sigma)$
$\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{c\}$
Return ϕ

procedure Signcrypt(m):
$c \xleftarrow{\$} \mathsf{Enc}(m, pk_R)$
$\sigma \xleftarrow{\$} \mathsf{Sign}(c, sk_S)$
$\phi \leftarrow (c, \sigma)$
$\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{(m, c)\}$
Return ϕ

procedure Unsigncrypt(ϕ):
$(c, \sigma) \leftarrow \phi$
If $c \in \mathcal{S}_1$ Return $\perp$
Find $(m, c) \in \mathcal{S}_2 \mid \mathsf{Verify}(c, \sigma, pk_S) = \mathsf{T}$
  Return m
If Not Found
  If $\mathsf{Verify}(c, \sigma, pk_S)$ Return $\mathsf{Dec}(c, sk_R)$
  Else Return $\perp$

procedure Finalize(b'):
Return $(b = b')$

Figure A.1: Game_0 defines IND-ogCCA2 for $\mathcal{E}t\mathcal{S}$

[1]The decryption-respecting relation f is such that $f((c_1, \sigma_1), (c_2, \sigma_2)) = \mathsf{T} \Leftrightarrow c_1 = c_2$.

Let E be the event where the unsigncryption oracle calls the second Verify and it returns T. The adversary $\mathcal{A}$ interacts with Game_1 (Figure A.2) exactly the same way it interacts with Game_0 (Figure A.1) unless the event E occurs. This transition is based on the failure event E, and as a result of the Difference Lemma (see Section 2.2), $|\Pr[S_1] - \Pr[S_0]| \leq \Pr[E]$.

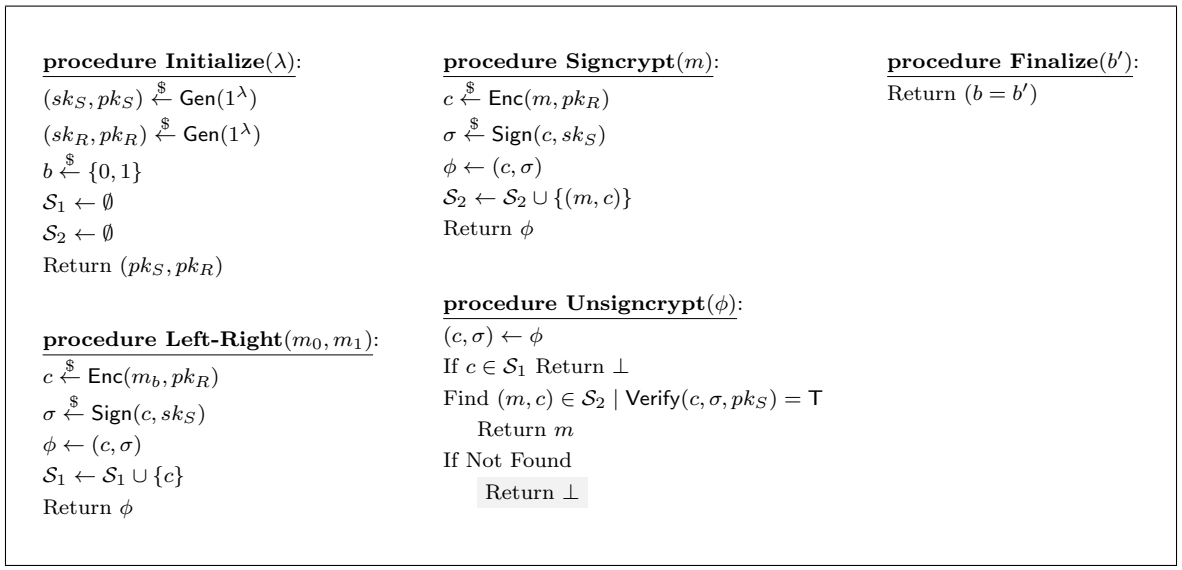

Figure A.2: Game_1

Intuitively, in order to make the event E occur, $\mathcal{A}$ has to break the security of the signature $\mathcal{S}$, which is $\mathsf{UF\text{-}CMA}$. We will argue this by building a program $\mathcal{B}$ (Figure A.3) that simulates the environment of Game_0 (Figure A.1) and wins $\mathsf{UF\text{-}CMA}$ (Figure 3.5) whenever event E occurs. Thus, $\Pr[E] = \mathbf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}CMA}}(\mathcal{B})$.

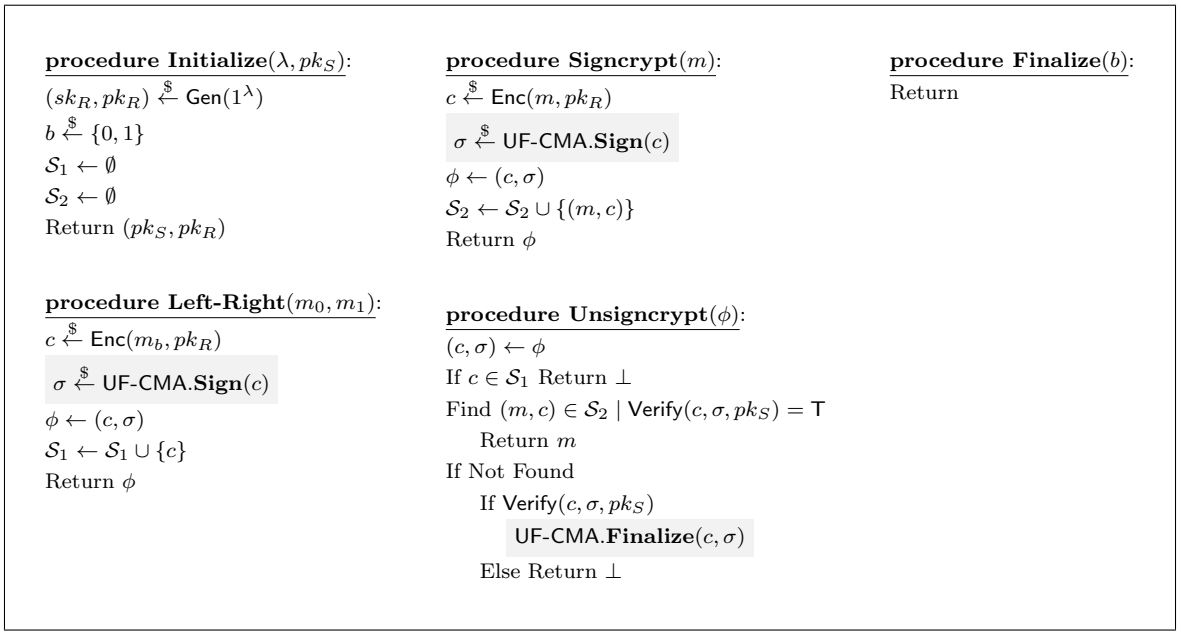

Figure A.3: Program $\mathcal{B}$ (Lemma 2.1)

Notice that in Game_1 (Figure A.2) the unsigncryption oracle is completely useless to the adversary $\mathcal{A}$ since it only returns $\perp$ or messages used by the adversary to query the signcryption oracle. We can easily construct a program $\mathcal{C}$ (Figure A.4) that simulates the environment of Game_1 and wins IND-CPA (Figure 3.1) when $\mathcal{A}$ wins Game_1.

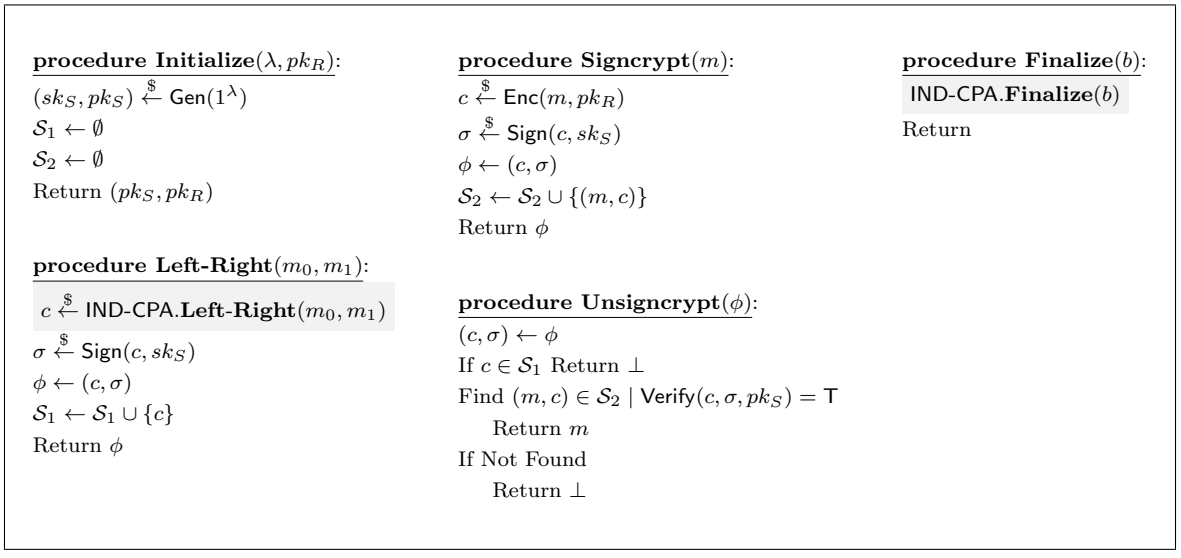

Figure A.4: Program $\mathcal{C}$ (Lemma 2.1)

Therefore: $\mathbf{Adv}^{\mathsf{IND\text{-}ogCCA2}}_{\mathcal{E}t\mathcal{S}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{S}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E}}(\mathcal{C})$.

□

Lemma 2.2. *If $\mathcal{E}$ is* IND-CPA *and $\mathcal{S}$ is* UF-CMA*, then $\mathcal{E}t\mathcal{S}$ is* UF-iCMA*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 (Figure A.5), which is game UF-iCMA expanded according to the $\mathcal{E}t\mathcal{S}$ construction. We prove the lemma by constructing a program $\mathcal{B}$ (Figure A.6) that simulates Game_0, and whenever $\mathcal{A}$ wins Game_0, $\mathcal{B}$ forges a signature on game UF-CMA (Figure 3.5).

procedure Initialize(λ):
$(sk_S, pk_S) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$\mathcal{S} \leftarrow \emptyset$
Return (pk_S, sk_R, pk_R)

procedure Signcrypt(m):
$c \xleftarrow{\$} \mathsf{Enc}(m, pk_R)$
$\sigma \xleftarrow{\$} \mathsf{Sign}(c, sk_S)$
$\phi \leftarrow (c, \sigma)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{m\}$
Return ϕ

procedure Finalize(ϕ):
$(c, \sigma) \leftarrow \phi$
If $\mathsf{Verify}(c, \sigma, pk_S)$ Then
  $m \leftarrow \mathsf{Dec}(c, sk_R)$
  If $m \neq \perp \wedge m \notin \mathcal{S}$ Return T
Return F

Figure A.5: Game_0 (defines UF-iCMA for $\mathcal{E}t\mathcal{S}$)

procedure Initialize(pk_S):
$(sk_R, pk_R) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
Return (pk_S, sk_R, pk_R)

procedure Signcrypt(m):
$c \xleftarrow{\$} \mathsf{Enc}(m, pk_R)$
$\sigma \xleftarrow{\$}$ UF-CMA.**Sign**(c)
$\phi \leftarrow (c, \sigma)$
Return ϕ

procedure Finalize(ϕ):
$(c, \sigma) \leftarrow \phi$
UF-CMA.**Finalize**(c, σ)
Return

Figure A.6: Program $\mathcal{B}$ (Lemma 2.2)

Therefore: $\mathbf{Adv}^{\mathsf{UF\text{-}iCMA}}_{\mathcal{E}t\mathcal{S}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{UF\text{-}CMA}}_{\mathcal{S}}(\mathcal{B})$.

□

A.2 Proof of theorem 3

The theorem follows from lemmas 3.1 and 3.2.

Lemma 3.1. *If $\mathcal{E}$ is* IND-gCCA2[2] *and $\mathcal{S}$ is* UF-NMA*, then $\mathcal{S}t\mathcal{E}$ is* UF-oCMA*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time adversary against Game_0 defined in Figure A.7, which is game UF-oCMA expanded according to the $\mathcal{S}t\mathcal{E}$ construction.

[2] $\mathcal{E}$ is IND-gCCA2 with respect to any decryption-respecting relation f.

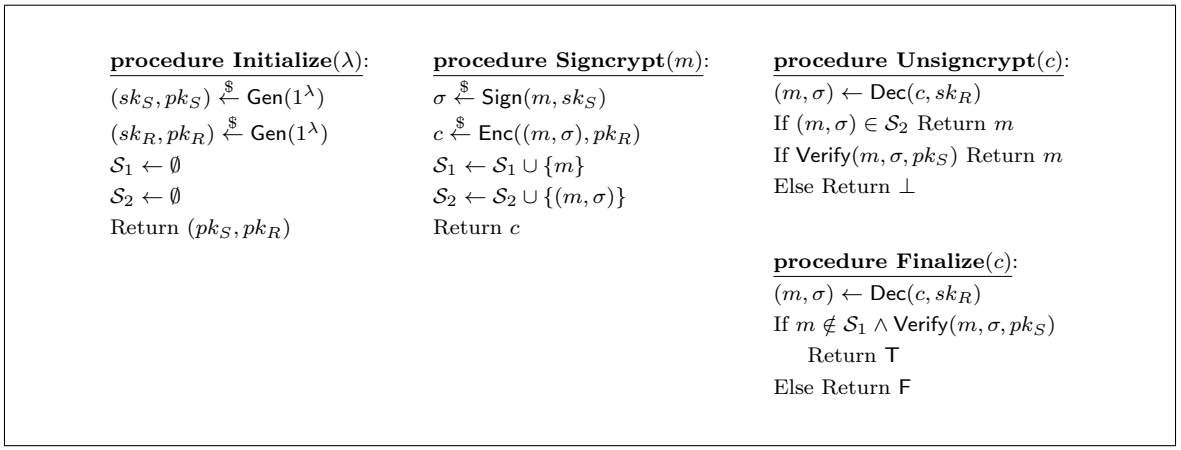

Figure A.7: Game_0 (defines UF-oCMA for $\mathcal{S}t\mathcal{E}$)

In Game_1 (Figure A.8), instead of correctly computing the signature of the message for the simulation of the signcryption oracle, the challenger samples a new ephemeral key and signs the message with this fake key. This is a transition based on indistinguishability. Intuitively, $\mathcal{A}$ cannot detect this change (except with negligible probability) because the signature is then encrypted (together with the message) and $\mathcal{E}$ is IND-gCCA2.

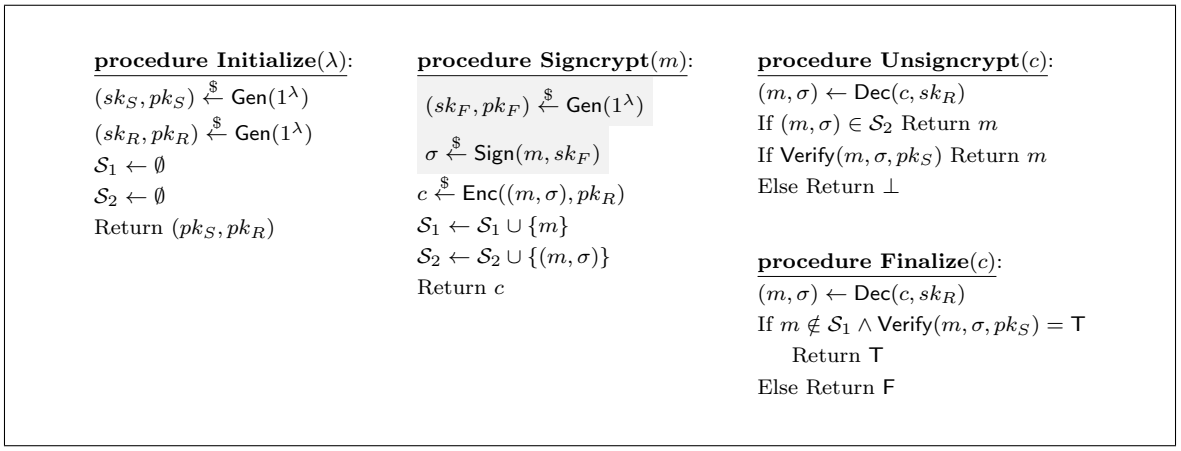

Figure A.8: Game_1

By building a distinguishing program $\mathcal{B}$ that interpolates between Game_0 and Game_1, we will argue that $\Pr[S_1] - \Pr[S_0] = \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}gCCA2\text{-}q}}(\mathcal{B})$. Game IND-gCCA2-q is identical to game IND-gCCA2 defined in Figure 4.1, expect that the adversary is allowed to retrieve multiple related ciphertexts by making q queries to the left-or-right oracle, chosen adaptively. Program $\mathcal{B}$ operates according to the description in Figure A.9.

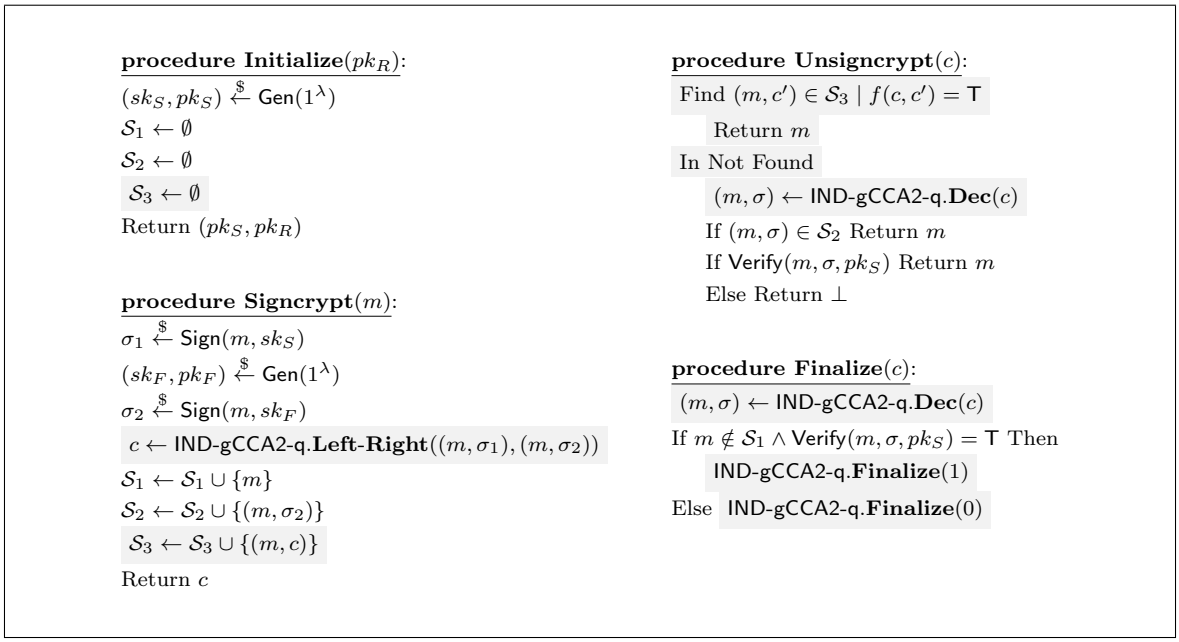

Figure A.9: Program $\mathcal{B}$ (Lemma 3.1)

The claim follows once we observe that:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{IND\text{-}gCCA2}}_{\mathcal{E}}(\mathcal{B}) &\stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{IND\text{-}gCCA2}\ \textit{returns}\ \mathsf{T}] - 1 \\
&= 2 \cdot \Pr[\mathcal{B}\ \textit{outputs}\ 0 \wedge b = 0] + 2 \cdot \Pr[\mathcal{B}\ \textit{outputs}\ 1 \wedge b = 1] - 1 \\
&= \Pr[\mathcal{B}\ \textit{outputs}\ 0 \mid b = 0] + \Pr[\mathcal{B}\ \textit{outputs}\ 1 \mid b = 1] - 1 \\
&= (1 - \Pr[S_0]) + \Pr[S_1] - 1 \\
&= \Pr[S_1] - \Pr[S_0]
\end{aligned}$$

Note that when $b = 0$, $\mathcal{B}$ simulates Game_0, and when $b = 1$, $\mathcal{B}$ simulates Game_1.
We now construct a program $\mathcal{C}$ which simulates the environment of Game_1 for $\mathcal{A}$, and each time $\mathcal{A}$ wins Game_1, $\mathcal{C}$ wins game UF-NMA as described in Figure 3.4. Consequently, $\Pr[S_1] = \mathbf{Adv}^{\mathsf{UF\text{-}NMA}}_{\mathcal{S}}(\mathcal{C})$. Program $\mathcal{C}$ is described in Figure A.10.

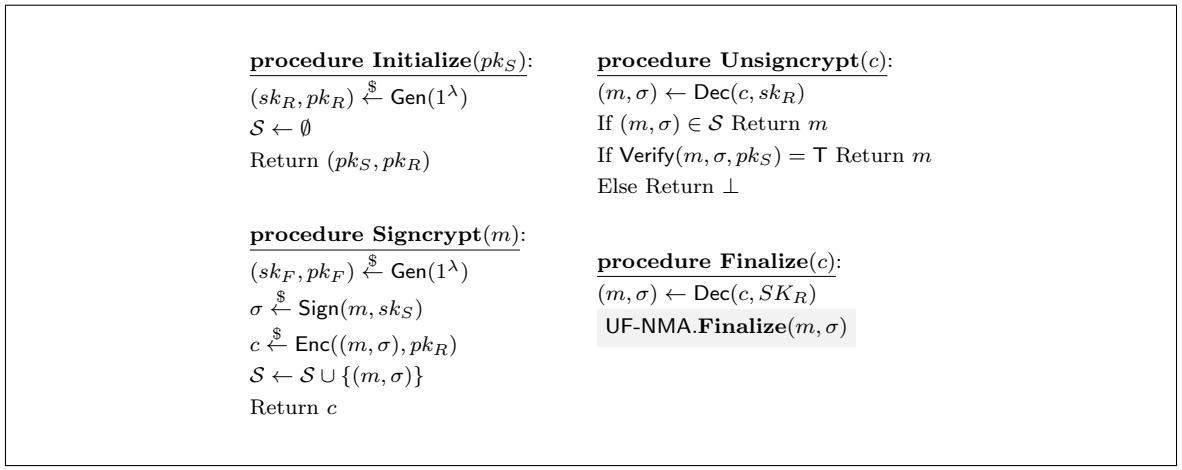

Figure A.10: Program $\mathcal{C}$

Using a hybrid argument, we conclude that an adversary making at most q queries to the left-or-right oracle cannot achieve an advantage greater than q times that of an adversary making just one query to the left-or-right oracle. Indeed, there exists a $\mathcal{D}$ such that $\mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}gCCA2\text{-}q}}(\mathcal{B}) \leq q \cdot \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}gCCA2}}(\mathcal{D})$. This argument is described in Section 5.4; simply consider that no information about the randomness is issued prior to the phase where the adversary queries the left-or-right oracle.
As a result, we have that: $\mathbf{Adv}_{\mathcal{S}t\mathcal{E}}^{\mathsf{UF\text{-}oCMA}}(\mathcal{A}) \leq q \cdot \mathbf{Adv}_{\mathcal{E}}^{\mathsf{IND\text{-}gCCA2}}(\mathcal{D}) + \mathbf{Adv}_{\mathcal{S}}^{\mathsf{UF\text{-}NMA}}(\mathcal{C})$.

□

Lemma 3.2. *If $\mathcal{E}$ is* IND-gCCA2 *and $\mathcal{S}$ is* UF-NMA*, then $\mathcal{S}t\mathcal{E}$ is* IND-igCCA2[3]*.*

Proof. Let $\mathcal{A}$ be any probabilistic polynomial-time (insider) adversary against Game_0 defined in Figure A.11, which is game IND-igCCA2 expanded according to the $\mathcal{S}t\mathcal{E}$ construction. We prove this lemma by building a program $\mathcal{B}$ (Figure A.12) that simulates the environment of Game_0 for $\mathcal{A}$ in a way that $\mathcal{B}$ wins IND-gCCA2 (Figure 4.1) when $\mathcal{A}$ wins Game_0.

[3] $\mathcal{S}t\mathcal{E}$ is IND-igCCA2 with respect to the same decryption-respecting relation f of $\mathcal{E}$.

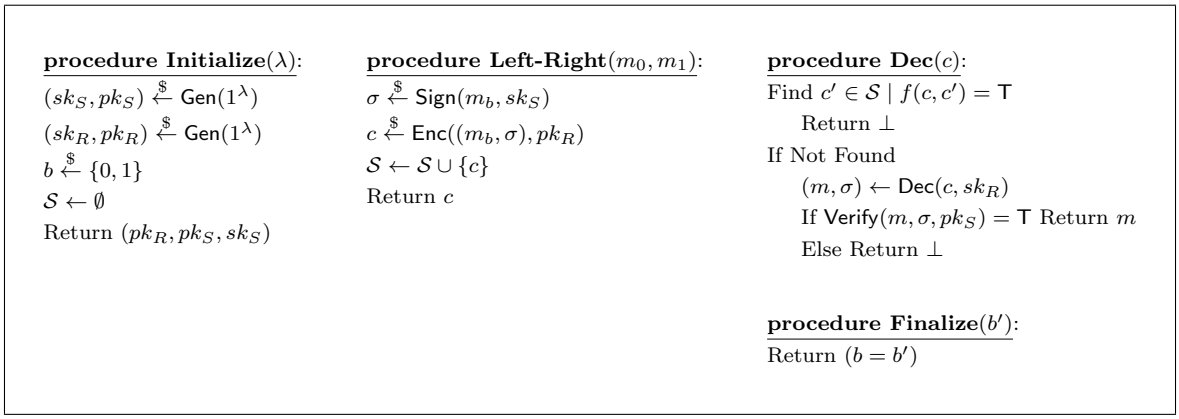

Figure A.11: Game_0 (defines IND-igCCA2 for $\mathcal{S}t\mathcal{E}$)

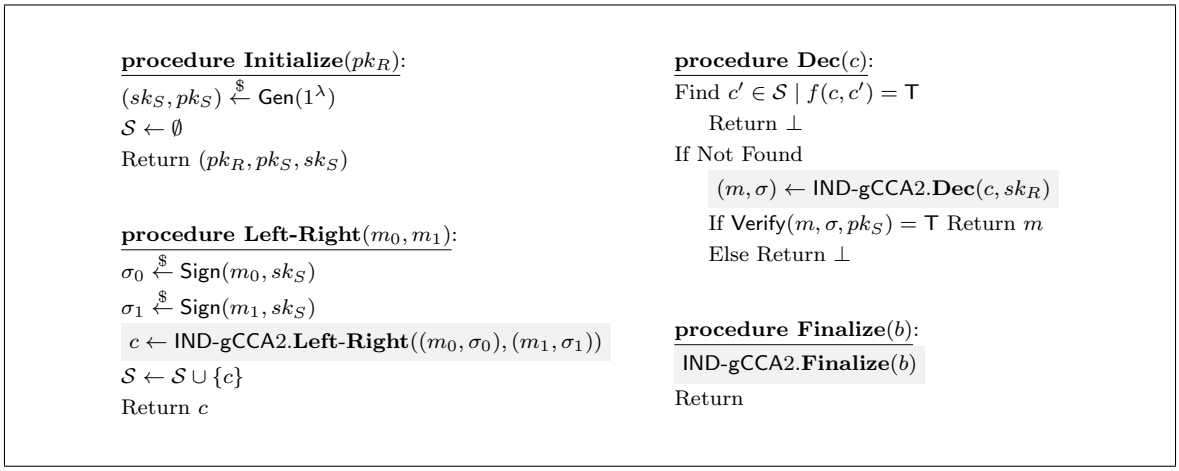

Figure A.12: Program $\mathcal{B}$ (Lemma 3.2)

Thus, $\mathbf{Adv}^{\mathsf{IND\text{-}igCCA2}}_{\mathcal{S}t\mathcal{E}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{IND\text{-}gCCA2}}_{\mathcal{E}}(\mathcal{B})$.

□

Bibliography

[1] FIPS 180-3. Secure hash standard. Technical report, National Institute of Standards and Technology — NIST, 2008.

[2] ISO/IEC 18033-2:2006. Information technology – security techniques – encryption algorithms – part 2: Asymmetric ciphers. Technical report, International Organization for Standardization — ISO, 2006.

[3] FIPS 197. Advanced encryption standard (aes). Technical report, National Institute of Standards and Technology — NIST, 2001.

[4] Special Publication 800-38A. Recommendation for block cipher modes of operation. Technical report, National Institute of Standards and Technology — NIST, 2001.

[5] Jee An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer Berlin / Heidelberg, 2002.

[6] Feng Bao and Robert Deng. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography — PKC '98*, volume 1431 of *Lecture Notes in Computer Science*, pages 55–59. Springer Berlin / Heidelberg, 1998.

[7] Manuel Barbosa and Pooya Farshim. Certificateless signcryption. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 369–372. ACM, 2008.

[8] Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemes. In *Public Key Cryptography — PKC 2003*,

volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer Berlin / Heidelberg, 2002.

[9] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer Berlin / Heidelberg, 1995.

[10] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology — EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer Berlin / Heidelberg, 2006.

[11] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *Advances in Cryptology — CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 537–554. Springer Berlin / Heidelberg, 2006.

[12] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21:149–177, 2008.

[13] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[14] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer Berlin / Heidelberg, 1998.

[15] Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[16] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[17] Alexander Dent. Hybrid signcryption schemes with insider security. In *Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 253–266. Springer Berlin / Heidelberg, 2005.

[18] Alexander Dent. Hybrid signcryption schemes with outsider security. In *Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 203–217. Springer Berlin / Heidelberg, 2005.

[19] Alexander Dent. A note on game-hopping proofs. Cryptology ePrint Archive: Report 2006/260, 2006.

[20] William Ehrsam, Carl Meyer, John Smith, and Walter Tuchman. Message verification and transmission error detection by block chaining. *US Patent 4074066*, 1976.

[21] Taher ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin, 1985.

[22] Marc Fischlin. The cramer-shoup strong-rsa signature scheme revisited. In *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer Berlin / Heidelberg, 2002.

[23] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. *Journal of Cryptology*, 17:81–104, 2004.

[24] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer Berlin / Heidelberg, 1999.

[25] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM, 1982.

[26] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[27] Jonathan Katz. *Digital Signatures*. Springer, 1st edition, 2010.

[28] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 1st edition, 2007.

[29] Neal Koblitz and Alfred J. Menezes. Another look at provable security. *Journal of Cryptology*, 20(1):3–37, 2007.

[30] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 345–359. Springer Berlin / Heidelberg, 2004.

[31] John Malone-Lee and Wenbo Mao. Two birds one stone: Signcryption using rsa. In *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 211–226. Springer Berlin / Heidelberg, 2003.

[32] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437. ACM, 1990.

[33] Charles Rackoff and Daniel Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Berlin / Heidelberg, 1992.

[34] Phillip Rogaway. On the role of definitions in and beyond cryptography. In *Advances in Computer Science — ASIAN 2004*, volume 3321 of *Lecture Notes in Computer Science*, pages 3252–3253. Springer Berlin / Heidelberg, 2005.

[35] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[36] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer Berlin / Heidelberg, 2000.

[37] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive: Report 2004/332, 2004.

[38] Ron Steinfeld and Yuliang Zheng. A signcryption scheme based on integer factorization. In *Information Security*, volume 1975 of *Lecture Notes in Computer Science*, pages 131–146. Springer Berlin / Heidelberg, 2000.

[39] PKCS #5 v2.0. Password-based cryptography standard. Technical report, RSA Laboratories, 1999.

[40] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer Berlin / Heidelberg, 2005.

[41] Dae Yum and Pil Lee. New signcryption schemes based on kcdsa. In *Information Security and Cryptology — ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 3–57. Springer Berlin / Heidelberg, 2002.

[42] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin / Heidelberg, 1997.

[43] Yuliang Zheng and Hideki Imai. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68(5):227–233, 1998.