Advanced Course on Contemporary Cryptology.
Advanced Courses CRM Barcelona, pages 133–189, June 2005.
ISBN: 3-7643-7294-X.

Contemporary Cryptology
Provable Security for Public Key Schemes

David Pointcheval

CNRS – LIENS, École normale supérieure, Paris, France.

Abstract. Since the appearance of public-key cryptography in the Diffie-Hellman seminal paper, many schemes have been proposed, but many have been broken. Indeed, for a long time, the simple fact that a cryptographic algorithm had withstood cryptanalytic attacks for several years was considered as a kind of validation. But some schemes took a long time before being widely studied, and maybe thereafter being broken.
A much more convincing line of research has tried to provide "provable" security for cryptographic protocols, in a complexity theory sense: if one can break the cryptographic protocol, one can efficiently solve the underlying problem. Unfortunately, this initially was a purely theoretical work: very few practical schemes could be proven in this so-called "standard model" because such a security level rarely meets with efficiency. Ten years ago, Bellare and Rogaway proposed a trade-off to achieve some kind of validation of efficient schemes, by identifying some concrete cryptographic objects with ideal random ones. The most famous identification appeared in the so-called "random-oracle model". More recently, another direction has been taken to prove the security of efficient schemes in the standard model (without any ideal assumption) by using stronger computational assumptions.
In these lectures, we focus on practical asymmetric protocols together with their "reductionist" security proofs, mainly in the random-oracle model. We cover the two main goals that public-key cryptography is devoted to solve: authentication with digital signatures, and confidentiality with public-key encryption schemes.

1 Introduction

Since the beginning of public-key cryptography, with the seminal Diffie-Hellman paper [25], many suitable algorithmic problems for cryptography have been proposed and many cryptographic schemes have been designed, together with more or less heuristic proofs of their security relative to the intractability of the above problems. However, most of those schemes have thereafter been broken.

The simple fact that a cryptographic algorithm withstood cryptanalytic attacks for several years has often been considered as a kind of validation procedure, but some schemes take a long time before being broken. An example is the Chor-Rivest cryptosystem [21, 48], based on the knapsack problem, which took more than 10 years to be totally broken [86], whereas before this attack it was believed to be strongly secure. As a consequence, the lack of attacks at some time should never be considered as a security validation of the proposal.

1.1 Provable Security

A completely different paradigm is provided by the concept of "provable" security. A significant line of research has tried to provide proofs in the framework of complexity theory (*a.k.a.* "reductionist" security proofs [4]): the proofs provide reductions from a well-studied problem (RSA or the discrete logarithm) to an attack against a cryptographic protocol.

At the beginning, people just tried to define the security notions required by actual cryptographic schemes, and then to design protocols which achieve these notions. The techniques were directly derived from the complexity theory, providing polynomial reductions. However, their aim was essentially theoretical. They were indeed trying to minimize the required assumptions on the primitives (one-way functions or permutations, possibly trapdoor, etc) [37,

35, 52, 71] without considering practicality. Therefore, they just needed to design a scheme with polynomial algorithms, and to exhibit polynomial reductions from the basic assumption on the primitive into an attack of the security notion, in an asymptotic way. However, such a result has no practical impact on actual security. Indeed, even with a polynomial reduction, one may be able to break the cryptographic protocol within a few hours, whereas the reduction just leads to an algorithm against the underlying problem which requires many years. Therefore, those reductions only prove the security when very huge (and thus maybe unpractical) parameters are in use, under the assumption that no polynomial time algorithm exists to solve the underlying problem.

1.2 Exact Security and Practical Security

For a few years, more efficient reductions have been expected, under the denominations of either "exact security" [12] or "concrete security" [58], which provide more practical security results. The perfect situation is reached when one manages to prove that, from an attack, one can describe an algorithm against the underlying problem, with almost the same success probability within almost the same amount of time. We have then achieved "practical security".

Unfortunately, in many cases, even just provable security is at the cost of an important loss in terms of efficiency for the cryptographic protocol. Thus some models have been proposed, trying to deal with the security of efficient schemes: some concrete objects are identified with ideal (or black-box) ones.

For example, it is by now usual to identify hash functions with ideal random functions, in the so-called "random-oracle model", informally introduced by Fiat and Shamir [28], and formalized by Bellare and Rogaway [10]. Similarly, block ciphers are identified with families of truly random permutations in the "ideal cipher model" [9]. A few years ago, another kind of idealization was introduced in cryptography, the black-box group [53, 80], where the group operation, in any algebraic group, is defined by a black-box: a new element necessarily comes from the addition (or the subtraction) of two already known elements. It is by now called the "generic model". Recent works [77, 18] even require several ideal models together to provide some new validations.

1.3 Outline of the Notes

In the next section, we explain and motivate more about exact security proofs, and we introduce the notion of the weaker security analyses, the security arguments (in an ideal model, and namely the random-oracle model). Then, we review the formalism of the most important asymmetric primitives: signatures and public-key encryption schemes. For both, we provide some examples, with some security analyses in the "reductionist" sense.

1.4 Related Work

These notes present a survey, based on several published papers, from the author, with often several co-authors: about signature [67, 69, 68, 17, 84], encryption [7, 3, 62, 59, 32, 33] and provably secure constructions [61, 63, 65, 64, 66]. Many other papers are also cited and rephrased, which present efficient provably secure constructions. Among the bibliography list presented at the end, we would like to insist on [10–12, 22, 82, 83]. We thus refer the reader to the original papers for more details.

2 Security Proofs and Security Arguments

2.1 Computational Assumptions

In both symmetric and asymmetric scenarios, many security notions can not be unconditionally guaranteed (whatever the computational power of the adversary). Therefore, security generally relies on a computational assumption: the existence of one-way functions, or permutations, possibly trapdoor. A one-way function is a function f which anyone can easily compute, but given $y = f(x)$ it is computationally intractable to recover x (or any pre-image of y). A one-way permutation is a bijective one-way function. For encryption, one would like the inversion to be possible for the recipient only: a trapdoor one-way permutation is a one-way permutation for which a secret information (the trapdoor) helps to invert the function on any point.

Given such objects, and thus computational assumptions about the intractability of the inversion without possible trapdoors, we would like that security could be achieved without extra assumptions. The only way to formally prove such a fact is by showing that an attacker against the cryptographic protocol can be used as a sub-part in an algorithm that can break the basic computational assumption.

A partial order therefore exists between computational assumptions (and intractable problems too): if a problem P is more difficult than the problem P' (P' reduces to P, see below) then the assumption of the intractability of the problem P is weaker than the assumption of the intractability of the problem P'. The weaker the required assumption is, the more secure the cryptographic scheme is.

2.2 "Reductionist" Security Proofs

In complexity theory, such an algorithm which uses the attacker as a sub-part in a global algorithm is called a reduction. If this reduction is polynomial, we can say that the attack of the cryptographic protocol is at least as hard as inverting the function: if one has a polynomial algorithm to solve the latter problem, one can polynomially solve the former one. In the complexity theory framework, a *polynomial* algorithm is the formalization of *efficiency*.

Therefore, in order to prove the security of a cryptographic protocol, one first needs to make precise the security notion one wants the protocol to achieve: which adversary's goal one wants to be intractable, under which kind of attack. At the beginning of the 1980's, such security notions have been defined for encryption [35] and signature [37, 38], and provably secure schemes have been suggested. However, those proofs had a theoretical impact only, because both the proposed schemes and the reductions were completely unpractical, yet polynomial. The reductions were indeed efficient (*i.e.* polynomial), and thus a polynomial attack against a cryptosystem would have led to a polynomial algorithm that broke the computational assumption. But the latter algorithm, even polynomial, may require hundreds of years to solve a small instance.

For example, let us consider a cryptographic protocol based on integer factoring. Let us assume that one provides a polynomial reduction from the factorization into an attack. But such a reduction may just lead to a factorization algorithm with a complexity in $2^{25}k^{10}$, where k is the bit-size of the integer to factor. This indeed contradicts the assumption that no-polynomial algorithm exists for factoring. However, on a 1024-bit number ($k = 2^{10}$), it provides an algorithm that requires 2^{125} basic operations, which is much more than the complexity of the best current algorithm, such as NFS [46], which needs less than 2^{100} (see Section 4). Therefore, such a reduction would just be meaningful for numbers above 4096 bits (since with $k = 2^{12}$,

$2^{145} < 2^{149}$, where 2^{149} is the estimate effort for factoring a 4096-bit integer with the best algorithm.) Concrete examples are given later.

2.3 Practical Security

Moreover, most of the proposed schemes were unpractical as well. Indeed, the protocols were polynomial in time and memory, but not efficient enough for practical implementation.

For a few years, people have tried to provide both practical schemes, with practical reductions and exact complexity, which prove the security for realistic parameters, under a well-defined assumption: exact reduction in the standard model (which means in the complexity-theoretic framework). For example, under the assumption that a 1024-bit integer cannot be factored with less than 2^{70} basic operations, the cryptographic protocol cannot be broken with less than 2^{60} basic operations. We will see such an example later.

Unfortunately, as already remarked, practical or even just efficient reductions in the standard model can rarely be conjugated with practical schemes. Therefore, one needs to make some hypotheses on the adversary: the attack is generic, independent of the actual implementation of some objects

- hash functions, in the "random-oracle model";
- symmetric block ciphers, in the "ideal-cipher model";
- algebraic groups, in the "generic model".

The "random-oracle model" was the first to be introduced in the cryptographic community [28, 10], and has already been widely accepted. By the way, flaws have been shown in the "generic model" [84] on practical schemes, and the "random-oracle model" is not equivalent to the standard one either. Several gaps have already been exhibited [19, 54, 6]. However, all the counter-examples in the random-oracle model are pathological, counter-intuitive and not natural. Therefore, in the sequel, we focus on security analyses in this model, for real and natural constructions. A security proof in the random-oracle model will at least give a strong argument in favor of the security of the scheme. Furthermore, proofs in the random-oracle model under a weak computational assumption may be of more pratical interest than proofs in the standard model under a strong computational assumption.

2.4 The Random-Oracle Model

As said above, efficiency rarely meets with provable security. More precisely, none of the most efficient schemes in their category have been proven secure in the standard model. However, some of them admit security validations under ideal assumptions: the random-oracle model is the most widely accepted one.

Many cryptographic schemes use a hash function $\mathcal{H}$ (such as MD5 [72] or the American standards SHA-1 [56], SHA-256, SHA-384 and SHA-512 [57]). This use of hash functions was originally motivated by the wish to sign long messages with a single short signature. In order to achieve *non-repudiation*, a minimal requirement on the hash function is the impossibility for the signer to find two different messages providing the same hash value. This property is called *collision-resistance*.

It was later realized that hash functions were an essential ingredient for the security of, first, signature schemes, and then of most cryptographic schemes. In order to obtain security arguments, while keeping the efficiency of the designs that use hash functions, a few authors suggested using the hypothesis that $\mathcal{H}$ behaves like a random function. First, Fiat and Shamir [28]

applied it heuristically to provide a signature scheme "as secure as" factorization. Then, Bellare and Rogaway [10–12] formalized this concept for cryptography, and namely for signature and public-key encryption.

In this model, the so-called "random-oracle model", the hash function can be formalized by an oracle which produces a truly random value for each new query. Of course, if the same query is asked twice, identical answers are obtained. This is precisely the context of relativized complexity theory with "oracles," hence the name.

About this model, no one has ever been able to provide a convincing contradiction to its practical validity, but just theoretical counter-examples on either clearly wrong designs for practical purpose [19], or artificial security notions [54, 6]. Therefore, this model has been strongly accepted by the community, and is considered as a good one, in which security analyses give a good taste of the actual security level. Even if it does not provide a formal proof of security (as in the standard model, without any ideal assumption), it is argued that proofs in this model ensure security of the overall design of the scheme provided that the hash function has no weakness, hence the name "security arguments".

This model can also be seen as a restriction on the adversary's capabilities. Indeed, it simply means that the attack is generic without considering any particular instantiation of the hash functions. Therefore, an actual attack would necessarily use a weakness or a specific feature of the hash function. The replacement of the hash function by another one would rule out this attack.

On the other hand, assuming the tamper-resistance of some devices, such as smart cards, the random-oracle model is equivalent to the standard model, which simply requires the existence of pseudo-random functions [34, 51].

As a consequence, almost all the standards bodies by now require designs provably secure, at least in that model, thanks to the security validation of very efficient protocols.

2.5 The General Framework

Before going into more details of this kind of proofs, we would like to insist on the fact that in the current general framework, we give the adversary complete access to the cryptographic primitive, but as a black-box. It can ask any query of its choice, and the box always answers correctly, in constant time. Such a model does not consider timing attacks [44], where the adversary tries to extract the secrets from the computational time. Some other attacks analyze the electrical energy required by a computation to get the secrets [45], or to make the primitive fail on some computation [13, 16]. They are not captured either by this model.

3 A First Formalism

In this section we describe more formally what a signature scheme and an encryption scheme are. Moreover, we make precise the security notions one wants the schemes to achieve. This is the first imperative step towards provable security.

3.1 Digital Signature Schemes

Digital signature schemes are the electronic version of handwritten signatures for digital documents: a user's signature on a message m is a string which depends on m, on public and secret data specific to the user and —possibly— on randomly chosen data, in such a way that anyone

can check the validity of the signature by using public data only. The user's public data are called the *public key*, whereas his secret data are called the *private key*. The intuitive security notion would be the impossibility to forge user's signatures without the knowledge of his private key. In this section, we give a more precise definition of signature schemes and of the possible attacks against them (most of those definitions are based on [38]).

Definitions A signature scheme $\mathsf{S} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined by the three following algorithms:

- The *key generation algorithm* $\mathcal{K}$. On input 1^k, which is a formal notation for a machine with running time polynomial in k (1^k is indeed k in basis 1), the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic. The input k is called the security parameter. The sizes of the keys, or of any problem involved in the cryptographic scheme, will depend on it, in order to achieve an appropriate security level (the expected minimal time complexity of any attack).
- The *signing algorithm* $\mathcal{S}$. Given a message m and a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$, $\mathcal{S}$ produces a signature σ. The signing algorithm might be probabilistic.
- The *verification algorithm* $\mathcal{V}$. Given a signature σ, a message m and a public key pk, $\mathcal{V}$ tests whether σ is a valid signature of m with respect to pk. In general, the verification algorithm need not be probabilistic.

Forgeries and Attacks In this subsection, we formalize some security notions which capture the main practical situations. On the one hand, the **goals** of the adversary may be various:

- Disclosing the private key of the signer. It is the most serious attack. This attack is termed *total break*.
- Constructing an efficient algorithm which is able to sign messages with good probability of success. This is called *universal forgery*.
- Providing a new message-signature pair. This is called *existential forgery*. The corresponding security level is called *existential unforgeability* (EUF).

In many cases the latter forgery, the *existential forgery*, is not dangerous because the output message is likely to be meaningless. Nevertheless, a signature scheme which is existentially forgeable does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys. Furthermore, it cannot formally guarantee the non-repudiation property, since anyone may be able to produce a message with a valid signature.

On the other hand, various **means** can be made available to the adversary, helping it into its forgery. We focus on two specific kinds of attacks against signature schemes: the *no-message attacks* and the *known-message attacks* (KMA). In the former scenario, the attacker only knows the public key of the signer. In the latter, the attacker has access to a list of valid message-signature pairs. According to the way this list was created, we usually distinguish many subclasses, but the strongest is definitely the *adaptive chosen-message attack* (CMA), where the attacker can ask the signer to sign any message of its choice, in an adaptive way: it can adapt its queries according to previous answers.

When signature generation is not deterministic, there may be several signatures corresponding to a given message. And then, some notions defined above may become ambiguous [84]. First, in known-message attacks, an existential forgery becomes the ability to forge a fresh message/signature pair that has not been obtained during the attack. There is a subtle point

here, related to the context where several signatures may correspond to a given message. We actually adopt the stronger rule that the attacker needs to forge the signature of message, whose signature was not queried. The more liberal rule, which makes the attacker successful when it outputs a second signature of a given message different from a previously obtained signature of the same message, is called *malleability*, while the corresponding security level is called *non-malleability* (NM). Similarly, in adaptive chosen-message attacks, the adversary may ask several times the same message, and each new answer gives it some information. A slightly weaker security model, by now called *single-occurrence adaptive chosen-message attack* (SO-CMA), allows the adversary at most one signature query for each message. In other words the adversary cannot submit the same message twice for signature.

When one designs a signature scheme, one wants to computationally rule out at least existential forgeries, or even achieve non-malleability, under adaptive chosen-message attacks. More formally, one wants that the success probability of any adversary $\mathcal{A}$ with a reasonable time is small, where

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathsf{S}}(\mathcal{A}) = \Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m,\sigma) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{sk}}}(\mathsf{pk}) : \mathcal{V}(\mathsf{pk},m,\sigma) = 1\right].$$

We remark that since the adversary is allowed to play an adaptive chosen-message attack, the signing algorithm is made available, without any restriction, hence the oracle notation $\mathcal{A}^{\mathcal{S}_{\mathsf{sk}}}$. Of course, in its answer, there is the natural restriction that, at least, the returned message-signature has not been obtained from the signing oracle $\mathcal{S}_{\mathsf{sk}}$ itself (non-malleability) or even the output message has not been queried (existential unforgeability).

3.2 Public-Key Encryption

The aim of a public-key encryption scheme is to allow anybody who knows the public key of Alice to send her a message that she will be the only one able to recover, granted her private key.

Definitions A public-key encryption scheme $\mathsf{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by the three following algorithms:

- The *key generation algorithm* $\mathcal{K}$. On input 1^k where k is the security parameter, the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk},\mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
- The *encryption algorithm* $\mathcal{E}$. Given a message m and a public key pk, $\mathcal{E}$ produces a ciphertext c of m. This algorithm may be probabilistic. In the latter case, we write $\mathcal{E}_{\mathsf{pk}}(m;r)$ where r is the random input to $\mathcal{E}$.
- The *decryption algorithm* $\mathcal{D}$. Given a ciphertext c and the private key sk, $\mathcal{D}_{\mathsf{sk}}(c)$ gives back the plaintext m. This algorithm is necessarily deterministic.

Security Notions As for signature schemes, the **goals** of the adversary may be various. The first common security notion that one would like for an encryption scheme is *one-wayness* (OW): with just public data, an attacker cannot get back the whole plaintext of a given ciphertext. More formally, this means that for any adversary $\mathcal{A}$, its success in inverting $\mathcal{E}$ without the private key should be negligible over the probability space $\mathbf{M} \times \Omega$, where $\mathbf{M}$ is the message

space and Ω is the space of the random coins r used for the encryption scheme, and the internal random coins of the adversary:

$$\mathsf{Succ}^{\mathsf{ow}}_{\mathsf{S}}(\mathcal{A}) = \Pr_{m,r}[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k) : \mathcal{A}(\mathsf{pk}, \mathcal{E}_{\mathsf{pk}}(m;r)) = m].$$

However, many applications require more from an encryption scheme, namely the *semantic security* (IND) [35], *a.k.a. polynomial security/indistinguishability of encryptions*: if the attacker has some information about the plaintext, for example that it is either "yes" or "no" to a crucial query, any adversary should not learn more with the view of the ciphertext. This security notion requires computational impossibility to distinguish between two messages, chosen by the adversary, which one has been encrypted, with a probability significantly better than one half: its advantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{S}}(\mathcal{A})$, formally defined as

$$2 \times \Pr_{b,r}\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{pk}),\\ c = \mathcal{E}_{\mathsf{pk}}(m_b;r) : \mathcal{A}_2(m_0, m_1, s, c) = b\end{array}\right] - 1,$$

where the adversary $\mathcal{A}$ is seen as a 2-stage attacker $(\mathcal{A}_1, \mathcal{A}_2)$, should be negligible.

A later notion is *non-malleability* (NM) [26]. To break it, the adversary, given a ciphertext, tries to produce a new ciphertext such that the plaintexts are meaningfully related. This notion is stronger than the above semantic security, but it is equivalent to the latter in the most interesting scenario [7] (the CCA attacks, see below). Therefore, we will just focus on one-wayness and semantic security.

On the other hand, an attacker can play many kinds of attacks, according to the **available information**: since we are considering asymmetric encryption, the adversary can encrypt any plaintext of its choice, granted the public key, hence the *chosen-plaintext attack* (CPA). It may furthermore have access to additional information, modeled by partial or full access to some oracles:

- A validity-checking oracle which, on input a ciphertext c, answers whether it is a valid ciphertext or not. Such a weak oracle, involved in the so-called reaction attacks [39] or *Validity-Checking Attack* (VCA), had been enough to break some famous encryption schemes [15, 42].
- A plaintext-checking oracle which, on input a pair (m, c), answers whether c encrypts the message m. This attack has been termed the *Plaintext-Checking Attack* (PCA) [59].
- The decryption oracle itself, which on any ciphertext answers the corresponding plaintext. There is of course the natural restriction not to ask the challenge ciphertext to that oracle.

For all these oracles, access may be restricted as soon as the challenge ciphertext is known, the attack is thus said *non-adaptive* since oracle queries cannot depend on the challenge ciphertext, while they depend on previous answers. On the opposite, access can be unlimited and attacks are thus called *adaptive attacks* (w.r.t. the challenge ciphertext). This distinction has been widely used for the chosen-ciphertext attacks, for historical reasons: the *non-adaptive chosen-ciphertext attacks* (CCA1) [52], *a.k.a.* lunchtime attacks, and *adaptive chosen-ciphertext attacks* (CCA2) [71]. The latter scenario which allows adaptively chosen ciphertexts as queries to the decryption oracle is definitely the strongest attack, and will be named the *chosen-ciphertext attack* (CCA).

Furthermore, multi-user scenarios can be considered where related messages are encrypted under different keys to be sent to many people (*e.g.* broadcast of encrypted data). This may provide many useful data for an adversary. For example, RSA is well-known to be weak in such a

scenario [40, 79], namely with a small encryption exponent, because of the Chinese Remainders Theorem. But once again, semantic security has been shown to be the appropriate security level, since it automatically extends to the multi-user setting: if an encryption scheme is semantically secure in the classical sense, it is also semantically secure in multi-user scenarios, against both passive [3] and active [5] adversaries.

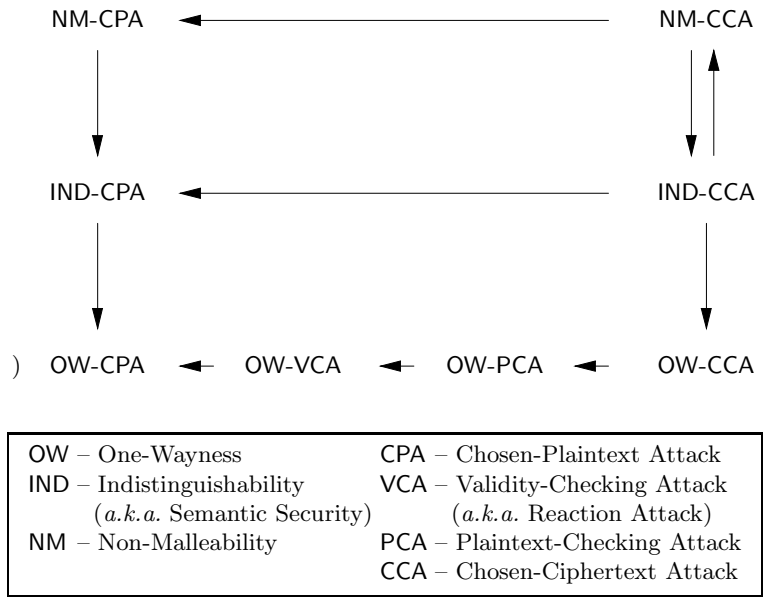

Fig. 1. Relations between the Security Notions for Asymmetric Encryption

A general study of these security notions and attacks was conducted in [7], we therefore refer the reader to this paper for more details. See also the summary diagram on Figure 1. However, we can just review the main scenarios we will consider in the following:

- one-wayness under chosen-plaintext attacks (OW-CPA) – where the adversary wants to recover the whole plaintext from just the ciphertext and the public key. This is the weakest scenario.
- semantic security under adaptive chosen-ciphertext attacks (IND-CCA) – where the adversary just wants to distinguish which plaintext, between two messages of its choice, has been encrypted, while it can ask any query it wants to a decryption oracle (except the challenge ciphertext). This is the strongest scenario one can define for encryption (still in our general framework.) Thus, this is our goal when we design a cryptosystem.

4 The Computational Assumptions

There are two major families in number theory-based public-key cryptography:

1. the schemes based on integer factoring, and on the RSA problem [73];
2. the schemes based on the discrete logarithm problem, and on the Diffie-Hellman problems [25], in any "suitable" group. The first groups in use were cyclic subgroups of $\mathbb{Z}_p^\star$, the multiplicative group of the modular quotient ring $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. But many schemes are now converted on cyclic subgroups of elliptic curves, or of the Jacobian of hyper-elliptic curves, with namely the so-called ECDSA [1], the US Digital Signature Standard [55] on elliptic curves.

4.1 Integer Factoring and the RSA Problem

The most famous intractable problem is factorization of integers: while it is easy to multiply two prime integers p and q to get the product $n = p \cdot q$, it is not simple to decompose n into its prime factors p and q.

Currently, the most efficient algorithm is based on sieving on number fields. The Number Field Sieve (NFS) method [46] has a super-polynomial, but sub-exponential, complexity in $\mathcal{O}(\exp((1.923 + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}))$. It has been used to establish the main record, in august 1999, by factoring a 155-digit integer (512 bits), product of two 78-digit primes [20]. The factored number, called RSA-155, was taken from the "RSA Challenge List", which is used as a yardstick for the security of the RSA cryptosystem (see later). The latter is used extensively in hardware and software to protect electronic data traffic such as in the SSL (Security Sockets Layer) Handshake Protocol.

This record is very important since 155 digits correspond to 512 bits. And this is the size which is in use in almost all the implementations of the RSA cryptosystem (namely for actual implementations of SSL on the Internet).

```
RSA-155 =
  109417386415705274218097073220403576120\
  037329454492059909138421314763499842889\
  347847179972578912673324976257528997818\
  337970765372440271467435315933543338 97
= 102639592829741105772054196573991675900\
  716567808038066803341933521790711307779
* 106603488380168454820927220360012878679\
  207958575989291522270608237193062808643
```

Unfortunately, integer multiplication just provides a one-way function, without any possibility to invert the process. No information is known to make factoring easier. However, some algebraic structures are based on the factorization of an integer n, where some computations are difficult without the factorization of n, but easy with it: the finite quotient ring $\mathbb{Z}_n$ which is isomorphic to the product ring $\mathbb{Z}_p \times \mathbb{Z}_q$ if $n = p \cdot q$.

For example, the *e-th* power of any element x can be easily computed using the *square-and-multiply* method. It consists in using the binary representation of the exponent $e = e_k e_{k-1} \dots e_0$, computing the successive 2 powers of x (x^{2^0}, x^{2^1}, $\dots$, x^{2^k}) and eventually to multiply altogether the ones for which $e_i = 1$. However, to compute *e-th* roots, it seems that one requires to know an integer d such that $ed = 1 \bmod \varphi(n)$, where $\varphi(n)$ is the totient Euler function which denotes the cardinality of the multiplicative subgroup $\mathbb{Z}_n^\star$ of $\mathbb{Z}_n$. In the particular case where $n = pq$, $\varphi(n) = (p-1)(q-1)$. And therefore, $ed - 1$ is a multiple of $\varphi(n)$ which is equivalent to the knowledge of the factorization of n [50]. In 1978, Rivest, Shamir and Adleman [73] defined the following problem:

> **The RSA Problem.** Let $n = pq$ be the product of two large primes of similar size and e an integer relatively prime to $\varphi(n)$. For a given $y \in \mathbb{Z}_n^\star$, compute the modular *e-th* root x of y (*i.e.* $x \in \mathbb{Z}_n^\star$ such that $x^e = y \bmod n$.)

The Euler function can be easily computed from the factorization of n, since for any $n = \prod p_i^{v_i}$,

$$\varphi(n) = n \times \prod \left(1 - \frac{1}{p_i}\right).$$

Therefore, with the factorization of n (the trapdoor), the RSA problem can be easily solved. But nobody knows whether the factorization is required, and how to do without it either:

> **The RSA Assumption.** For any product of two primes, $n = pq$, large enough, the RSA problem is intractable (presumably as hard as the factorization of n).

4.2 The Discrete Logarithm and the Diffie-Hellman Problems

The setting is quite general: one is given

- a cyclic group $\mathcal{G}$ of prime order q (such as the finite group $(\mathbb{Z}_q, +)$, a subgroup of $(\mathbb{Z}_p^\star, \times)$ for $q|p-1$, of an elliptic curve, etc);
- a generator $\mathbf{g}$ (*i.e.* $\mathcal{G} = \langle \mathbf{g} \rangle$).

We note in bold (such as $\mathbf{g}$) any element of the group $\mathcal{G}$, to distinguish it from a scalar $x \in \mathbb{Z}_q$. But such a $\mathbf{g}$ could be an element in $\mathbb{Z}_p^\star$ or a point of an elliptic curve, according to the setting. Above, we talked about a "suitable" group $\mathcal{G}$. In such a group, some of the following problems have to be hard to solve (using the additive notation).

- the **Discrete Logarithm** problem (**DL**): given $\mathbf{y} \in \mathcal{G}$, compute $x \in \mathbb{Z}_q$ such that $\mathbf{y} = x \cdot \mathbf{g} = \mathbf{g} + \ldots + \mathbf{g}$ (x times), then one writes $x = \log_{\mathbf{g}} \mathbf{y}$.
- the **Computational Diffie-Hellman** problem (**CDH**): given two elements in the group $\mathcal{G}$, $\mathbf{a} = a \cdot \mathbf{g}$ and $\mathbf{b} = b \cdot \mathbf{g}$, compute $\mathbf{c} = ab \cdot \mathbf{g}$. Then one writes $\mathbf{c} = \mathbf{DH}(\mathbf{a}, \mathbf{b})$.
- the **Decisional Diffie-Hellman** Problem (**DDH**): given three elements in the group $\mathcal{G}$, $\mathbf{a} = a \cdot \mathbf{g}$, $\mathbf{b} = b \cdot \mathbf{g}$ and $\mathbf{c} = c \cdot \mathbf{g}$, decide whether $\mathbf{c} = \mathbf{DH}(\mathbf{a}, \mathbf{b})$ (or equivalently, whether $c = ab \bmod q$).

It is clear that they are sorted from the strongest problem to the weakest one. Furthermore, one may remark that they all are "random self-reducible", which means that any instance can be reduced to a uniformly distributed instance: for example, given a specific element $\mathbf{y}$ for which one wants to compute the discrete logarithm x in basis $\mathbf{g}$, one can choose a random $z \in \mathbb{Z}_q$, and compute $\mathbf{z} = z \cdot \mathbf{y}$. The element $\mathbf{z}$ is therefore uniformly distributed in the group, and the discrete logarithm $\alpha = \log_{\mathbf{g}} \mathbf{z}$ leads to $x = \alpha / z \bmod q$. As a consequence, there are only average complexity cases. Thus, the ability to solve a problem for a non-negligible fraction of instances in polynomial time is equivalent to solve any instance in expected polynomial time.

A new variant of the Diffie-Hellman problem has more recently been defined by Tatsuaki Okamoto and the author [60], the so-called *Gap Diffie-Hellman Problem* (**GDH**), where one wants to solve the **CDH** problem with an access to a **DDH** oracle. One may easily remark the following properties about above problems: $\mathbf{DL} \geq \mathbf{CDH} \geq \{\mathbf{DDH}, \mathbf{GDH}\}$, where $A \geq B$ means that the problem A is at least as hard as the problem B. However, in practice, no one knows how to solve any of them without breaking the **DL** problem itself.

Currently, the most efficient algorithms to solve the latter problem depend on the underlying group. For generic groups (for which no specific algebraic property can be used), algorithms have a complexity in the square root of q, the order of the generator $\mathbf{g}$ [78, 70]. For example, on well-chosen elliptic curves only these algorithms can be used. The last record was established in April 2001 on the curve defined by the equation $y^2 + xy = x^3 + x^2 + 1$ over the finite field with 2^{109} elements.

However, for subgroups of $\mathbb{Z}_p^\star$, some better techniques can be applied. The best algorithm is based on sieving on number fields, as for the factorization. The General Number Field Sieve method [41] has a super-polynomial, but sub-exponential, complexity in $\mathcal{O}(\exp((1.923 +$

$o(1))(\ln p)^{1/3}(\ln\ln p)^{2/3}))$. It was used to establish the last record, in April 2001 as well, by computing discrete logarithms in $\mathbb{Z}_p^\star$, for a 120-digit prime p. Therefore, 512-bit primes are still safe enough, as far as the generic attacks cannot be used (the generator must be of large order q, at least a 160-bit prime)

For signature applications, one only requires groups where the **DL** problem is hard, whereas encryption needs trapdoor problems and therefore requires groups where some of the **DH**'s problems are also hard to solve.

5 Digital Signature Schemes

Until 1996, no practical **DL**-based cryptographic scheme has ever been formally studied, but heuristically only. And surprisingly, at the Eurocrypt '96 conference, two opposite studies were conducted on the El Gamal signature scheme [27], the first **DL**-based signature scheme designed in 1985 and depicted on Figure 2.

Initialization $\rightarrow (p, g)$
g a generator of $\mathbb{Z}_p^\star$, where p is a large prime $\rightarrow (p, g)$
$\mathcal{K}$**: Key Generation** $\rightarrow (y, x)$
private key $x \in \mathbb{Z}_{p-1}^\star$ public key $y = g^x \bmod p$ $\rightarrow (y, x)$
$\mathcal{S}$**: Signature of** $m \rightarrow (r, s)$
K is randomly chosen in $\mathbb{Z}_{p-1}^\star$ $r = g^K \bmod p \quad s = (m - xr)/K \bmod p-1$ $\rightarrow (r, s)$ is a signature of m
$\mathcal{V}$**: Verification of** (m, r, s)
check whether $g^m \stackrel{?}{=} y^r r^s \bmod p$ $\rightarrow$ Yes/No

Fig. 2. The El Gamal Signature Scheme.

Whereas existential forgeries were known for that scheme, it was believed to prevent universal forgeries. The first analysis, from Daniel Bleichenbacher [14], showed such a universal forgery when the generator g is not properly chosen. The second one, from Jacques Stern and the author [67], proved the security against existential forgeries under adaptive chosen-message attacks of a slight variant with a randomly chosen generator g. The latter variant simply replaces the message m by $\mathcal{H}(m, r)$ in the computation, while one uses a hash function $\mathcal{H}$ that is assumed to behave like a random oracle. It is amazing to remark that the Bleichenbacher's attack also applies on our variant. Therefore, depending on the initialization, our variant could be a very strong signature scheme or become a very weak one!

As a consequence, a proof has to be performed in details, with precise assumptions and achievements. Furthermore, the conclusions have to be strictly followed by developers, otherwise the concrete implementation of a secure scheme can be very weak.

5.1 Provable Security

The first *secure* signature scheme was proposed by Goldwasser *et al.* [37] in 1984. It used the notion of claw-free permutations. A pair of permutations (f, g) is said *claw-free* if it is

computationally impossible to find a *claw* (x, y), which satisfies $f(x) = g(y)$. Their proposal provided polynomial algorithms with a polynomial reduction between the research of a claw and an existential forgery under an adaptive chosen-message attack. However, the scheme was totally unpractical. What about practical schemes?

The RSA Signature Scheme Two years after the Diffie-Hellman paper [25], Rivest, Shamir and Adleman [73] proposed the first signature scheme based on the "trapdoor one-way permutation paradigm", using the RSA function: the generation algorithm produces a large composite number $N = pq$, a public key e, and a private key d such that $e \cdot d = 1 \bmod \varphi(N)$. The signature of a message m, encoded as an element in $\mathbb{Z}_N^\star$, is its *e-th* root, $\sigma = m^{1/e} = m^d \bmod N$. The verification algorithm simply checks whether $m = \sigma^e \bmod N$.

However, the RSA scheme is not secure by itself since it is subject to existential forgery: it is easy to create a valid message-signature pair, without any help of the signer, first randomly choosing a certificate σ and getting the signed message m from the public verification relation, $m = \sigma^e \bmod N$.

The Schnorr Signature Scheme In 1986 a new paradigm for signature schemes was introduced. It is derived from fair zero-knowledge identification protocols involving a prover and a verifier [36], and uses hash functions in order to create a kind of virtual verifier. The first application was derived from the Fiat–Shamir [28] zero-knowledge identification protocol, based on the hardness of extracting square roots, with a brief outline of its security. Another famous identification scheme [75], together with the signature scheme [76], has been proposed later by Schnorr, based on that paradigm: the generation algorithm produces two large primes p and q, such that $q \geq 2^k$, where k is the security parameter, and $q \,|\, p - 1$, as well as an element g in $\mathbb{Z}_p^\star$ of order q. It also creates a pair of keys, the private key $x \in \mathbb{Z}_q^\star$ and the public key $y = g^{-x} \bmod p$ The signature of a message m is a triple (r, e, s), where $r = g^K \bmod p$, with a random $K \in \mathbb{Z}_q$, the "challenge" $e = \mathcal{H}(m, r)$ and $s = K + ex \bmod q$. The latter satisfies $r = g^s y^e \bmod p$ with $e = \mathcal{H}(m, r)$, which is checked by the verification algorithm.

The security results for that paradigm have been considered as folklore for a long time but without any formal validation.

5.2 DL-Based Signatures

In our papers [67, 68], with Jacques Stern, we formally proved the above paradigm when $\mathcal{H}$ is assumed to behave like a random oracle. The proof is based on the by now classical *oracle replay technique*: by a polynomial replay of the attack with different random oracles (the $\mathcal{Q}_i$'s are the queries and the ρ_i's are the answers), we allow the attacker to forge signatures that are suitably related. This generic technique is depicted on Figure 3, where the signature of a

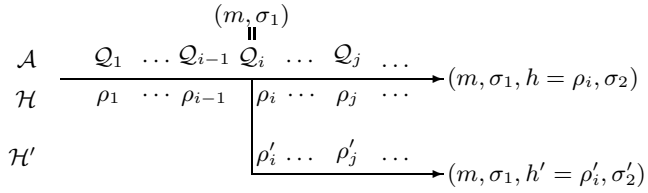

Fig. 3. The Oracle Replay Technique

message m is a triple (σ_1, h, σ_2), with $h = \mathcal{H}(m, \sigma_1)$ which depends on the message and the first

part of the signature, both bound not to change for the computation of σ_2, which really relies on the knowledge of the private key. If the probability of fraud is high enough, then with good probability, the adversary is able to answer to many distinct outputs from the $\mathcal{H}$ function, on the input (m, σ_1).

To be more concrete, let us consider the Schnorr signature scheme, which is presented on Figure 4, in any "suitable" cyclic group $\mathcal{G}$ of prime order q, where at least the Discrete Logarithm problem is hard. We expect to obtain two signatures $(\mathbf{r} = \sigma_1, h, s = \sigma_2)$ and $(\mathbf{r}' = \sigma'_1, h', s' = \sigma'_2)$

Initialization (security parameter k) $\rightarrow (\mathcal{G}, g, \mathcal{H})$
$\mathbf{g}$ a generator of any cyclic group $(\mathcal{G}, +)$ of order q, with $2^{k-1} \leq q < 2^k$ $\mathcal{H}$ a hash function: $\{0,1\}^\star \rightarrow \mathbb{Z}_q$ $\rightarrow (\mathcal{G}, g, \mathcal{H})$

$\mathcal{K}$**: Key Generation** $\rightarrow (\mathbf{y}, x)$
private key $x \in \mathbb{Z}_q^\star$ public key $\mathbf{y} = -x \cdot \mathbf{g}$ $\rightarrow (\mathbf{y}, x)$

$\mathcal{S}$**: Signature of** $m \rightarrow (\mathbf{r}, h, s)$
K is randomly chosen in $\mathbb{Z}_q^\star$ $\mathbf{r} = K \cdot \mathbf{g}$ $\quad h = \mathcal{H}(m, r)$ $\quad s = K + xh \bmod q$ $\rightarrow (\mathbf{r}, h, s)$ is a signature of m

$\mathcal{V}$**: Verification of** (m, r, s)
check whether $h \stackrel{?}{=} \mathcal{H}(m, \mathbf{r})$ and $\mathbf{r} \stackrel{?}{=} s \cdot \mathbf{g} + h \cdot \mathbf{y}$ $\rightarrow$ Yes/No

Fig. 4. The Schnorr Signature Scheme.

of an identical message m such that $\sigma_1 = \sigma'_1$, but $h \neq h'$. Thereafter, we can easily extract the discrete logarithm of the public key:

$$\left.\begin{array}{l} \mathbf{r} = s \cdot \mathbf{g} + h \cdot \mathbf{y} \\ \mathbf{r} = s' \cdot \mathbf{g} + h' \cdot \mathbf{y} \end{array}\right\} \Rightarrow (s - s') \cdot \mathbf{g} = (h' - h) \cdot \mathbf{y},$$

which leads to $\log_{\mathbf{g}} \mathbf{y} = (s - s') \cdot (h' - h)^{-1} \bmod q$.

General Tools First, let us recall the "Splitting Lemma" which will be the main probabilistic tool for the "Forking Lemma". It translates the fact that when a subset A is "large" in a product space $X \times Y$, it has many "large" sections.

Lemma 1 (The Splitting Lemma). *Let $A \subset X \times Y$ such that $\Pr[(x, y) \in A] \geq \varepsilon$. For any $\alpha < \varepsilon$, define*

$$B = \left\{ (x, y) \in X \times Y \mid \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \alpha \right\},$$

then the following statements hold:

(i) $\Pr[B] \geq \alpha$
(ii) $\forall (x, y) \in B, \Pr_{y' \in Y}[(x, y') \in A] \geq \varepsilon - \alpha$.
(iii) $\Pr[B \mid A] \geq \alpha / \varepsilon$.

Proof. In order to prove statement (i), we argue by contradiction, using the notation $\bar{B}$ for the complement of B in $X \times Y$. Assume that $\Pr[B] < \alpha$. Then

$$\varepsilon \leq \Pr[B] \cdot \Pr[A \mid B] + \Pr[\bar{B}] \cdot \Pr[A \mid \bar{B}] < \alpha \cdot 1 + 1 \cdot (\varepsilon - \alpha) = \varepsilon.$$

This implies a contradiction, hence the result.

Statement (ii) is a straightforward consequence of the definition.

We finally turn to the last assertion, using Bayes' law:

$$\begin{aligned}\Pr[B \mid A] &= 1 - \Pr[\bar{B} \mid A] \\ &= 1 - \Pr[A \mid \bar{B}] \cdot \Pr[\bar{B}] / \Pr[A] \geq 1 - (\varepsilon - \alpha)/\varepsilon = \alpha/\varepsilon.\end{aligned}$$

No-Message Attacks. The following *Forking Lemma* just states that the above oracle replay technique will often success with any good adversary.

Theorem 2 (The Forking Lemma). *Let $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme with security parameter k, with a signature as above, of the form $(m, \sigma_1, h, \sigma_2)$, where $h = \mathcal{H}(m, \sigma_1)$ and σ_2 depends on σ_1 and h only. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask q_h queries to the random oracle, with $q_h > 0$. We assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 7q_h/2^k$, a valid signature $(m, \sigma_1, h, \sigma_2)$. Then, within time $T' \leq 16q_hT/\varepsilon$, and with probability $\varepsilon' \geq 1/9$, a replay of this machine outputs two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$.*

Proof. We are given an adversary $\mathcal{A}$, which is a probabilistic polynomial time Turing machine with random tape ω. During the attack, this machine asks a polynomial number of questions to the random oracle $\mathcal{H}$. We may assume that these questions are distinct: for instance, $\mathcal{A}$ can store questions and answers in a table. Let $\mathcal{Q}_1, \ldots, \mathcal{Q}_{q_h}$ be the q_h distinct questions and let $\rho = (\rho_1, \ldots, \rho_{q_h})$ be the list of the q_h answers of $\mathcal{H}$. It is clear that a random choice of $\mathcal{H}$ exactly corresponds to a random choice of ρ. Then, for a random choice of $(\omega, \mathcal{H})$, with probability ε, $\mathcal{A}$ outputs a valid signature $(m, \sigma_1, h, \sigma_2)$. Since $\mathcal{H}$ is a random oracle, it is easy to see that the probability for h to be equal to $\mathcal{H}(m, \sigma_1)$ is less than $1/2^k$, unless it has been asked during the attack. So, it is likely that the question (m, σ_1) is actually asked during a successful attack. Accordingly, we define $Ind_{\mathcal{H}}(\omega)$ to be the index of this question: $(m, \sigma_1) = \mathcal{Q}_{Ind_{\mathcal{H}}(\omega)}$ (we let $Ind_{\mathcal{H}}(\omega) = \infty$ if the question is never asked). We then define the sets

$$\begin{aligned}\mathbf{S} &= \left\{(\omega, \mathcal{H}) \mid \mathcal{A}^{\mathcal{H}}(\omega) \text{ succeeds } \&\ Ind_{\mathcal{H}}(\omega) \neq \infty\right\}, \\ \text{and } \mathbf{S}_i &= \left\{(\omega, \mathcal{H}) \mid \mathcal{A}^{\mathcal{H}}(\omega) \text{ succeeds } \&\ Ind_{\mathcal{H}}(\omega) = i\right\} \quad \text{for } i \in \{1, \ldots, q_h\}.\end{aligned}$$

We thus call $\mathbf{S}$ the set of the successful pairs $(\omega, \mathcal{H})$.

One should note that the set $\{\mathbf{S}_i \mid i \in \{1, \ldots, q_h\}\}$ is a partition of $\mathbf{S}$. With those definitions, we find a lower bound for the probability of success, $\nu = \Pr[\mathbf{S}] \geq \varepsilon - 1/2^k$. Since we did the assumption that $\varepsilon \geq 7q_h/2^k \geq 7/2^k$, then $\nu \geq 6\varepsilon/7$. Let I be the set consisting of the most likely indices i,

$$I = \{i \mid \Pr[\mathbf{S}_i \mid \mathbf{S}] \geq 1/2q_h\}.$$

The following lemma claims that, in case of success, the index lies in I with probability at least $1/2$.

Lemma 3.

$$\Pr[Ind_{\mathcal{H}}(\omega) \in I \mid \mathbf{S}] \geq \frac{1}{2}.$$

Proof. By definition of the sets $\mathbf{S}_i$, $\Pr[Ind_{\mathcal{H}}(\omega) \in I \mid \mathbf{S}] = \sum_{i \in I} \Pr[\mathbf{S}_i \mid \mathbf{S}]$. This probability is equal to $1 - \sum_{i \notin I} \Pr[\mathbf{S}_i \mid \mathbf{S}]$. Since the complement of I contains fewer than q_h elements, this probability is at least $1 - q_h \times 1/2q_h \geq 1/2$.

We now run the attacker $2/\varepsilon$ times with random ω and random $\mathcal{H}$. Since $\nu = \Pr[\mathbf{S}] \geq 6\varepsilon/7$, with probability greater than $1 - (1 - 6\varepsilon/7)^{2/\varepsilon}$, we get at least one pair $(\omega, \mathcal{H})$ in $\mathbf{S}$. It is easily seen that this probability is lower bounded by $1 - e^{-12/7} \geq 4/5$.

We now apply the Splitting-lemma (Lemma 1, with $\varepsilon = \nu/2q_h$ and $\alpha = \varepsilon/2$) for each integer $i \in I$: we denote by $\mathcal{H}_{|i}$ the restriction of $\mathcal{H}$ to queries of index strictly less than i. Since $\Pr[\mathbf{S}_i] \geq \nu/2q_h$, there exists a subset Ω_i of executions such that,

$$\text{for any } (\omega, \mathcal{H}) \in \Omega_i, \Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathbf{S}_i \mid \mathcal{H}'_{|i} = \mathcal{H}_{|i}] \geq \frac{\nu}{4q_h}$$
$$\Pr[\Omega_i \mid \mathbf{S}_i] \geq \frac{1}{2}.$$

Since all the subsets $\mathbf{S}_i$ are disjoint,

$$\begin{aligned}
&\Pr_{\omega,\mathcal{H}}[(\exists i \in I)\ (\omega, \mathcal{H}) \in \Omega_i \cap \mathbf{S}_i \mid \mathbf{S}] \\
&\quad = \Pr\left[\bigcup_{i \in I} (\Omega_i \cap \mathbf{S}_i) \mid \mathbf{S}\right] = \sum_{i \in I} \Pr[\Omega_i \cap \mathbf{S}_i \mid \mathbf{S}] \\
&\quad = \sum_{i \in I} \Pr[\Omega_i \mid \mathbf{S}_i] \cdot \Pr[\mathbf{S}_i \mid \mathbf{S}] \geq \left(\sum_{i \in I} \Pr[\mathbf{S}_i \mid \mathbf{S}]\right)/2 \geq \frac{1}{4}.
\end{aligned}$$

We let β denote the index $Ind_{\mathcal{H}}(\omega)$ corresponding to the successful pair. With probability at least $1/4$, $\beta \in I$ and $(\omega, \mathcal{H}) \in \mathbf{S}_\beta \cap \Omega_\beta$. Consequently, with probability greater than $4/5 \times 1/5 = 1/5$, the $2/\varepsilon$ attacks have provided a successful pair $(\omega, \mathcal{H})$, with $\beta = Ind_{\mathcal{H}}(\omega) \in I$ and $(\omega, \mathcal{H}) \in \mathbf{S}_\beta$. Furthermore, if we replay the attack, with fixed ω but randomly chosen oracle $\mathcal{H}'$ such that $\mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}$, we know that $\Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathbf{S}_\beta \mid \mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}] \geq \nu/4q_h$. Then

$$\begin{aligned}
&\Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathbf{S}_\beta \text{ and } \rho_\beta \neq \rho'_\beta \mid \mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}] \\
&\quad \geq \Pr_{\mathcal{H}'}[(\omega, \mathcal{H}') \in \mathbf{S}_\beta \mid \mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}] - \Pr_{\mathcal{H}'}[\rho'_\beta = \rho_\beta] \geq \nu/4q_h - 1/2^k,
\end{aligned}$$

where $\rho_\beta = \mathcal{H}(\mathcal{Q}_\beta)$ and $\rho'_\beta = \mathcal{H}'(\mathcal{Q}_\beta)$. Using again the assumption that $\varepsilon \geq 7q_h/2^k$, the above probability is lower-bounded by $\varepsilon/14q_h$. We thus replay the attack $14q_h/\varepsilon$ times with a new random oracle $\mathcal{H}'$ such that $\mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}$, and get another success with probability greater than $1 - (1 - \varepsilon/14q_h)^{14q_h/\varepsilon} \geq 1 - e^{-1} \geq 3/5$.

Finally, after less than $2/\varepsilon + 14q_h/\varepsilon$ repetitions of the attack, with probability greater than $1/5 \times 3/5 \geq 1/9$, we have obtained two signatures $(m, \sigma_1, h, \sigma_2)$ and $(m', \sigma'_1, h', \sigma'_2)$, both valid w.r.t. their specific random oracle $\mathcal{H}$ or $\mathcal{H}'$, and with the particular relations

$$\mathcal{Q}_\beta = (m, \sigma_1) = (m', \sigma'_1) \text{ and } h = \mathcal{H}(\mathcal{Q}_\beta) \neq \mathcal{H}'(\mathcal{Q}_\beta) = h'.$$

One may have noticed that the mechanics of our reduction depend on some parameters related to the attacker $\mathcal{A}$, namely, its probability of success ε and the number q_h of queries to the random oracle. This induces a lack of uniformity. A uniform version, in expected polynomial time is also possible.

Theorem 4 (The Forking Lemma – The Uniform Case). *Let $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme with security parameter k, with a signature as above, of the form $(m, \sigma_1, h, \sigma_2)$, where $h = \mathcal{H}(m, \sigma_1)$ and σ_2 depends on σ_1 and h only. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask q_h queries to the random oracle, with $q_h > 0$. We assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 7q_h/2^k$, a valid signature $(m, \sigma_1, h, \sigma_2)$. Then there is another machine which has control over $\mathcal{A}$ and produces two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$, in expected time $T' \leq 84480Tq_h/\varepsilon$.*

Proof. Now, we try to design a machine $\mathbf{M}$ which succeeds in expected polynomial time:

1. $\mathbf{M}$ initializes $j = 0$;
2. $\mathbf{M}$ runs $\mathcal{A}$ until it outputs a successful pair $(\omega, \mathcal{H}) \in \mathbf{S}$ and denotes by N_j the number of calls to $\mathcal{A}$ to obtain this success, and by β the index $Ind_{\mathcal{H}}(\omega)$;
3. $\mathbf{M}$ replays, at most $140N_j\alpha^j$ times, $\mathcal{A}$ with fixed ω and random $\mathcal{H}'$ such that $\mathcal{H}'_{|\beta} = \mathcal{H}_{|\beta}$, where $\alpha = 8/7$;
4. $\mathbf{M}$ increments j and returns to 2, until it gets a successful forking.

For any execution of $\mathbf{M}$, we denote by J the last value of j and by N the total number of calls to $\mathcal{A}$. We want to compute the expectation of N. Since $\nu = \Pr[\mathbf{S}]$, and $N_j \geq 1$, then $\Pr[N_j \geq 1/5\nu] \geq 3/4$. We define $\ell = \lceil \log_\alpha q_h \rceil$, so that, $140N_j\alpha^j \geq 28q_h/\varepsilon$ for any $j \geq \ell$, whenever $N_j \geq 1/5\nu$. Therefore, for any $j \geq \ell$, when we have a first success in $\mathbf{S}$, with probability greater than $1/4$, the index $\beta = Ind_{\mathcal{H}}(\omega)$ is in the set I and $(\omega, \mathcal{H}) \in \mathbf{S}_\beta \cap \Omega_\beta$. Furthermore, with probability greater than $3/4$, $N_j \geq 1/5\nu$. Therefore, with the same conditions as before, that is $\varepsilon \geq 7q_h/2^k$, the probability of getting a successful fork after at most $28q_h/\varepsilon$ iterations at step 3 is greater than $6/7$.
For any $t \geq \ell$, the probability for J to be greater or equal to t is less than $(1 - 1/4 \times 3/4 \times 6/7)^{t-\ell}$, which is less than $\gamma^{t-\ell}$, with $\gamma = 6/7$. Furthermore,

$$E[N \mid J = t] \leq \sum_{j=0}^{j=t} \left(E[N_j] + 140E[N_j]\alpha^j \right) \leq \frac{141}{\nu} \times \sum_{j=0}^{j=t} \alpha^j \leq \frac{141}{\nu} \times \frac{\alpha^{t+1}}{\alpha - 1}.$$

So, the expectation of N is $E[N] = \sum_t E[N \mid J = t] \cdot \Pr[J = t]$ and then it can be shown to be less than $84480q_h/\varepsilon$. Hence the theorem.

Chosen-Message Attacks. However, this just covers the no-message attacks, without any oracle access. Since we can simulate any zero-knowledge protocol, even without having to restart the simulation because of the honest verifier (*i.e.* the challenge is randomly chosen by the random oracle $\mathcal{H}$) one can easily simulate the signer without the private key:

- one first chooses random $h, s \in \mathbb{Z}_q$;
- one computes $\mathbf{r} = s \cdot \mathbf{g} + h \cdot \mathbf{y}$ and defines $\mathcal{H}(m, \mathbf{r})$ to be equal to h, which is a uniformly distributed value;
- one can output $(\mathbf{r}, h, s)$ as a valid signature of the message m.

This furthermore simulates the oracle $\mathcal{H}$, by defining $\mathcal{H}(m, \mathbf{r})$ to be equal to h. This simulation is almost perfect since $\mathcal{H}$ is supposed to output a random value to any new query, and h is indeed a random value. Nevertheless, if the query $\mathcal{H}(m, \mathbf{r})$ has already been asked, $\mathcal{H}(m, \mathbf{r})$ is already defined, and thus the definition $\mathcal{H}(m, \mathbf{r}) \leftarrow h$ is impossible. But such a situation is

very rare, which allows us to claim the following result, which stands for the Schnorr signature scheme but also for any signature derived from a three-round honest verifier zero-knowledge interactive proof of knowledge:

Theorem 5. *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. We denote respectively by q_h and q_s the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of queries that $\mathcal{A}$ can ask to the signer. Assume that, within a time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 10(q_s+1)(q_s+q_h)/2^k$, a valid signature $(m, \sigma_1, h, \sigma_2)$. If the triples (σ_1, h, σ_2) can be simulated without knowing the secret key, with an indistinguishable distribution probability, then, a replay of the attacker $\mathcal{A}$, where interactions with the signer are simulated, outputs two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$, within time $T' \leq 23 q_h T/\varepsilon$ and with probability $\varepsilon' \geq 1/9$.*

A uniform version of this lemma can also be found in [68]. From a more practical point of view, these results state that if an adversary manages to perform an existential forgery under an adaptive chosen-message attack within an expected time T, after q_h queries to the random oracle and q_s queries to the signing oracle, then the discrete logarithm problem can be solved within an expected time less than Cq_hT, for some constant C. This result has been more recently extended to the transformation of any identification scheme secure against passive adversaries into a signature scheme [8].

Brickell, Vaudenay, Yung and the author also extended the *forking lemma* technique [69, 17] to many variants of El Gamal [27] and DSA [55], such as the Korean Standard KCDSA [43]. However, the original El Gamal and DSA schemes were not covered by this study, and are certainly not provably secure, even if no attack has ever been found against DSA.

5.3 RSA-Based Signatures

Unfortunately, with the above signatures based on the discrete logarithm, as any construction using the Fiat-Shamir paradigm, we do not really achieve our goal, because the reduction is costly, since q_h can be huge, as much as 2^{60} in practice. This security proof is meaningful for very large groups only.

FDH-RSA: The Full-Domain Hash Signature In 1996, Bellare and Rogaway [12] proposed other candidates, based on the RSA assumption. The first scheme is the by-now classical hash-and-decrypt paradigm (*a.k.a.* the Full-Domain Hash paradigm): as for the basic RSA signature, the generation algorithm produces a large composite number $N = pq$, a public key e, and a private key d such that $e \cdot d = 1 \bmod \varphi(N)$. In order to sign a message m, one first hashes it using a full-domain hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_N^\star$, and computes the *e-th* root, $\sigma = \mathcal{H}(m)^d \bmod N$. The verification algorithm simply checks whether the following equality holds, $\mathcal{H}(m) = \sigma^e \bmod N$.

More generally, the Full-Domain Hash signature can be defined as described on figure 5, for any trapdoor one-way permutation f.

Security Analysis For this scheme, Bellare and Rogaway proved, in the random-oracle model:

$\mathcal{K}$: **Key Generation** $\rightarrow (f, f^{-1})$
public key $f : X \longrightarrow X$, a trapdoor one-way permutation onto X private key f^{-1} $\rightarrow (f, f^{-1})$
$\mathcal{S}$: **Signature of** $m \rightarrow \sigma$
$r = \mathcal{H}(m)$ and $\sigma = f^{-1}(r)$ $\rightarrow \sigma$ is the signature of m
$\mathcal{V}$: **Verification of** (m, σ)
check whether $f(\sigma) \stackrel{?}{=} \mathcal{H}(m)$ $\rightarrow$ Yes/No

Fig. 5. The FDH Signature.

Theorem 6. *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within a time t, after q_h and q_s queries to the hash function and the signing oracle respectively. Then the permutation f can be inverted with probability ε' within time t' where*

$$\varepsilon' \geq \frac{\varepsilon}{q_s + q_h + 1} \qquad \text{and} \qquad t' \leq t + (q_s + q_h)T_f,$$

with T_f the time for an evaluation of f.

Let us present this proof, using the new formalism introduced by Victor Shoup in [81–83], and which will be extensively used in these notes. In this technique, we define a sequence $\mathbf{G}_1$, $\mathbf{G}_2$, etc., of modified attack games starting from the actual game $\mathbf{G}_0$. Each of the games operates on the same underlying probability space: the public and private keys of the cryptographic scheme, the coin tosses of the adversary $\mathcal{A}$ and the random oracles. Only the rules defining how the view is computed differ from game to game. To go from one game to another with a slightly different distribution probability, we repeatedly use the following lemma:

Lemma 7. *Let E_1, E_2 and F_1, F_2 be events defined on a probability space*

$$\Pr[\mathsf{E}_1 \mid \neg\mathsf{F}_1] = \Pr[\mathsf{E}_2 \mid \neg\mathsf{F}_2] \text{ and } \Pr[\mathsf{F}_1] = \Pr[\mathsf{F}_2] = \varepsilon \;\Rightarrow\; |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \varepsilon.$$

Proof. The proof follows from easy computations:

$$\begin{aligned}
|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| &= |\Pr[\mathsf{E}_1 \mid \mathsf{F}_1] \cdot \Pr[\mathsf{F}_1] + \Pr[\mathsf{E}_1 \mid \neg\mathsf{F}_1] \cdot \Pr[\neg\mathsf{F}_1] \\
&\quad - \Pr[\mathsf{E}_2 \mid \mathsf{F}_2] \cdot \Pr[\mathsf{F}_2] - \Pr[\mathsf{E}_2 \mid \neg\mathsf{F}_2] \cdot \Pr[\neg\mathsf{F}_2]| \\
&= |(\Pr[\mathsf{E}_1 \mid \mathsf{F}_1] - \Pr[\mathsf{E}_2 \mid \mathsf{F}_2]) \cdot \varepsilon \\
&\quad + (\Pr[\mathsf{E}_1 \mid \neg\mathsf{F}_1] - \Pr[\mathsf{E}_2 \mid \neg\mathsf{F}_2]) \cdot (1 - \varepsilon)| \\
&= |(\Pr[\mathsf{E}_1 \mid \mathsf{F}_1] - \Pr[\mathsf{E}_2 \mid \mathsf{F}_2]) \cdot \varepsilon| \leq \varepsilon.
\end{aligned}$$

Actually, this lemma will not be used in the proofs of the FDH signatures, because all the simulated distributions will remain perfect.

Basic Proof of the FDH *Signature.* In this proof, we incrementally define a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_5$. We make a very detailed sequence of games in this proof, since this is the first one. Some steps will be skipped in the other proofs. The goal of this proof is to reduce the inversion of the permutation f on an element y (find x such that $y = f(x)$) to an attack. We are thus given such a random challenge y.

Game $\mathbf{G}_0$: This is the real attack game, in the random-oracle model, which includes the verification step. This means that the attack game consists in giving the public key to the adversary, and a full access to the signing oracle. When it outputs its forgery, one furthermore checks whether it is actually valid or not. Note that if the adversary asks q_s queries to the signing oracle and q_h queries to the hash oracle, at most $q_s + q_h + 1$ queries are asked to the hash oracle during this game, since each signing query may make such a new query, and the last verification step too. We are interested in the following event: S_0 which occurs if the verification step succeeds (and the signature is new).

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathsf{fdh}}(\mathcal{A}) = \Pr[\mathsf{S}_0]. \tag{1}$$

Game $\mathbf{G}_1$: In this game, we simulate the oracles, the hash oracle $\mathcal{H}$ and the signing oracle $\mathcal{S}$, and the last verification step, as shown on Figure 6. From this simulation, we easily see that the game is perfectly indistinguishable from the real attack.

$$\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0]. \tag{2}$$

$\mathcal{H}$ oracle	For a hash-query $\mathcal{H}(q)$, such that a record $(q, \star, r)$ appears in H-List, the answer is r. Otherwise the answer r is defined according to the following rule: ►**Rule** $\mathcal{H}^{(1)}$ Choose a random element $r \in X$. The record $(q, \perp, r)$ is added to H-List. Note: the second component of the elements of this list will be explained later.
$\mathcal{S}$ oracle	For a sign-query $\mathcal{S}(m)$, one first asks for $r = \mathcal{H}(m)$ to the $\mathcal{H}$-oracle, and then the signature σ is defined according to the following rule: ►**Rule** $\mathcal{S}^{(1)}$ Computes $\sigma = f^{-1}(r)$.
$\mathcal{V}$ oracle	The game ends with the verification of the output (m, σ) from the adversary. One first asks for $r = \mathcal{H}(m)$, and checks whether $r = f(\sigma)$.

Fig. 6. Simulation of the Attack Game against FDH

Game $\mathbf{G}_2$: Since the verification process is included in the attack game, the output message is necessarily asked to the hash oracle. Let us guess the index c of this (first) query. If the guess failed, we abort the game. Therefore, only a correct guess (event GoodGuess) may lead to a success.

$$\begin{aligned}\Pr[\mathsf{S}_2] &= \Pr[\mathsf{S}_1 \wedge \mathsf{GoodGuess}] = \Pr[\mathsf{S}_1 \mid \mathsf{GoodGuess}] \times \Pr[\mathsf{GoodGuess}] \\ &\geq \Pr[\mathsf{S}_1] \times \frac{1}{q_h + q_s + 1}.\end{aligned} \tag{3}$$

Game $\mathbf{G}_3$: We can now simulate the hash oracle, incorporating the challenge y, for which we want to extract the pre-image x by f:

►**Rule** $\mathcal{H}^{(3)}$
> If this is the *c-th* query, set $r \leftarrow y$; otherwise, choose a random element $r \in X$. The record $(q, \perp, r)$ is added to H-List.

Because of the random choice for the challenge y, this rule lets the game indistinguishable from the previous one.

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2]. \tag{4}$$

Game $\mathbf{G}_4$: We now modify the simulation of the hash oracle for other queries, which may be used in signing queries:

►**Rule** $\mathcal{H}^{(4)}$
> If this is the *c-th* query, set $r \leftarrow y$ and $s \leftarrow \perp$; otherwise, choose a random element $s \in X$, and compute $r = f(s)$. The record (q, s, r) is added to H-List.

Because of the permutation property of f, and the random choice for s, this rule lets the game indistinguishable from the previous one.

$$\Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_3]. \tag{5}$$

Game $\mathbf{G}_5$: By now, excepted for the *c-th* hash query, which will be involved in the forgery (and thus not asked to the signing oracle), the pre-image is known. One can thus simulate the signing oracle without quering f^{-1}:

►**Rule** $\mathcal{S}^{(5)}$
> Lookup for (m, s, r) in H-List, and set $\sigma = s$.

Since the message corresponding to the *c-th* query cannot be asked to the signing oracle, otherwise it would not be a valid forgery, this rule lets the game indistinguishable from the previous one.

$$\Pr[\mathsf{S}_5] = \Pr[\mathsf{S}_4]. \tag{6}$$

Note that now, the simulation can easily be performed, without any specific computational power or oracle access. Just a few more evaluations of f are done to simulate the hash oracle, and the forgery leads to the pre-image of y:

$$\Pr[\mathsf{S}_5] = \mathsf{Succ}^{\mathsf{ow}}_f(t + (q_h + q_s)T_f). \tag{7}$$

As a consequence, using equations (1), (2), (3), (4), (5), (6) and (7)

$$\begin{aligned}\mathsf{Succ}^{\mathsf{ow}}_f(t + (q_h + q_s)T_f) = \Pr[\mathsf{S}_5] = \Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_2] \\ \geq \frac{1}{q_h + q_s + 1} \times \Pr[\mathsf{S}_1] \geq \frac{1}{q_h + q_s + 1} \times \Pr[\mathsf{S}_0].\end{aligned}$$

And thus,

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathsf{fdh}}(\mathcal{A}) \leq (q_h + q_s + 1) \times \mathsf{Succ}^{\mathsf{ow}}_f(t + (q_h + q_s)T_f).$$

□

Improved Security Result. This reduction has been thereafter improved [22], thanks to the random self-reducibility of the RSA function. The following result applies as soon as the one-way permutation has some homomorphic property on the group X:

$$f(x \otimes y) = f(x) \otimes f(y).$$

Theorem 8. *Let $\mathcal{A}$ be an adversary which can produce, with success probability ε, an existential forgery under a chosen-message attack within a time t, after q_h and q_s queries to the hash function and the signing oracle respectively. Then the permutation f can be inverted with probability ε' within time t' where*

$$\varepsilon' \geq \frac{\varepsilon}{q_s} \times \exp(-2) \quad \text{and} \quad t' \leq t + (q_s + q_h)T_f,$$

with T_f the time for an evaluation of f.

This proof can be performed as the previous one, and thus starts at the real game $\mathbf{G}_0$, then we can use the same simulation as in the game $\mathbf{G}_1$. The sole formal difference in the simulation will be the H-List which elements have one more field, and are thus initially of the form $(q, \perp, \perp, r)$. Things differ much after that, using a real value p between 0 and 1, which will be made precise later. The idea here, is to make any forgery useful for inverting the permutation f, not only a specific (guessed) one. On the other hand, one must still be able to simulate the signing oracle. The probability p will separate the two situations:

Game $\mathbf{G}_2$: A random coin decides whether we introduce the challenge y in the hash answer, or an element with a known pre-image:

►**Rule $\mathcal{H}^{(2)}$**
One chooses a random $s \in X$. With probability p, one sets $r \leftarrow y \otimes f(s)$ and $t \leftarrow 1$; otherwise, $r \leftarrow f(s)$ and $t \leftarrow 0$. The record (q, t, s, r) is added to H-List.

Because of the homomorphic property on the group X of the permutation f, this rule lets the game indistinguishable from the previous one. Note again that elements in H-List contain one more field t than in the previous proof. One may see that $r = y^t \otimes f(s)$.

Game $\mathbf{G}_3$: For a proportion $1 - p$ of the signature queries, one can simulate the signing oracle without having to invert the permutation f:

►**Rule $\mathcal{S}^{(3)}$**
Lookup for (m, t, s, r) in H-List, if $t = 1$ then halt the game, otherwise set $\sigma = s$.

This rule lets the game indistinguishable, unless one signing query fails ($t = 1$), which happens with probability p, for each signature:

$$\Pr[\mathsf{S}_3] = (1 - p)^{q_s} \times \Pr[\mathsf{S}_2]. \tag{8}$$

Note that now, the simulation can easily be performed, without any specific computational power or oracle access. Just a few more exponentiations are done to simulate the hash oracle, and the forgery (m, σ) leads to the pre-image of y, if $(t = 1)$. The latter case holds with probability p. Indeed, (m, t, s, r) can be found in the H-List, and then $r = y^t \otimes f(s) = y \otimes f(s) = f(\sigma)$, which easily leads to the pre-image of y by f:

$$\mathsf{Succ}^{\mathsf{ow}}_f(t + (q_h + q_s)T_f) = p \times \Pr[\mathsf{S}_3]. \tag{9}$$

Using equations (1), (2), (8) and (9)

$$\begin{aligned}\mathsf{Succ}^{\mathsf{ow}}_f(t+(q_h+q_s)T_f) &= p\times \Pr[\mathsf{S}_3] = p\times(1-p)^{q_s}\times\Pr[\mathsf{S}_2]\\ &= p\times(1-p)^{q_s}\times\Pr[\mathsf{S}_1] = p\times(1-p)^{q_s}\times\Pr[\mathsf{S}_0].\end{aligned}$$

And thus,

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathsf{fdh}}(\mathcal{A}) \le \frac{1}{p(1-p)^{q_s}} \times \mathsf{Succ}^{\mathsf{ow}}_f(t+(q_h+q_s)T_f).$$

Therefore, the success probability of our inversion algorithm is $p(1-p)^{q_s}\varepsilon$, if ε is the success probability of the adversary. If $q_s > 0$, the latter expression is optimal for $p = 1/(q_s+1)$. And for this parameter, and a huge value q_s, the success probability is approximately ε/eq_s. It is anyway larger than ε/e^2q_s (where $e = \exp(1) \approx 2.17\ldots$).

As far as time complexity is concerned, each random oracle simulation (which can be launched by a signing simulation) requires a modular exponentiation to the power e, hence the result. □

This is a great improvement since the success probability does not depend anymore on q_h. Furthermore, q_s can be limited by the user, whereas q_h cannot. In practice, one only assumes $q_h \le 2^{60}$, but q_s can be limited below 2^{30}.

The Probabilistic Signature Scheme However, one would like to get more, suppressing any coefficient. In their paper [12], Bellare and Rogaway proposed such a better candidate, the Probabilistic Signature Scheme (PSS, see Figure 7): the key generation is still the same, but

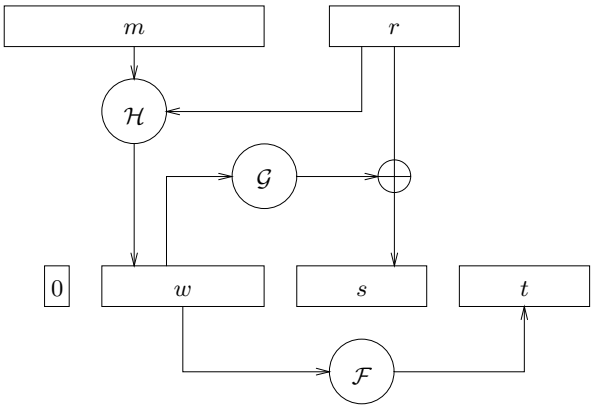

Fig. 7. Probabilistic Signature Scheme

the signature process involves three hash functions

$$\mathcal{F}:\{0,1\}^{k_2}\to\{0,1\}^{k_0},\quad \mathcal{G}:\{0,1\}^{k_2}\to\{0,1\}^{k_1},$$
$$\mathcal{H}:\{0,1\}^{\star}\to\{0,1\}^{k_2},$$

where $k = k_0+k_1+k_2+1$ satisfies $\{0,1\}^{k-1} \subset X \subset \{0,1\}^k$. We remind that f is a trapdoor one-way permutation onto X, with an homomorphic relationship. For each message m to be signed, one chooses a random string $r \in \{0,1\}^{k_1}$. One first computes $w = \mathcal{H}(m,r)$, $s = \mathcal{G}(w)\oplus r$ and $t = \mathcal{F}(w)$. Then one concatenates $y = 0\|w\|s\|t$, where $a\|b$ denotes the concatenation of the bit strings a and b. Finally, one computes the pre-image by f, $\sigma = f^{-1}(y)$. The verification

algorithm first computes $y = f(\sigma)$, and parses it as $y = b\|w\|s\|t$. Then, one can get $r = s \oplus \mathcal{G}(w)$, and checks whether $b = 0$, $w = \mathcal{H}(m, r)$ and $t = \mathcal{F}(w)$.

About this PSS construction, Bellare and Rogaway proved the security in the random-oracle model.

Theorem 9. *Let $\mathcal{A}$ be a CMA-adversary against f–PSS which produces an existential forgery within a time t, after q_f, q_g, q_h and q_s queries to the hash functions $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ and the signing oracle respectively. Then its success probability is upper-bounded by*

$$\mathsf{Succ}^{\mathsf{ow}}_f(t + (q_s + q_h)k_2 \cdot T_f) + \frac{1}{2^{k_2}} + (q_s + q_h) \cdot \left(\frac{q_s}{2^{k_1}} + \frac{q_f + q_g + q_h + q_s + 1}{2^{k_2}} \right),$$

with T_f the time for an evaluation of f.

Proof. First, we assume the existence of an adversary $\mathcal{A}$ that produces an existential forgery with probability ε within time t, after q_f, q_g and q_h queries to the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ and q_s queries to the signing oracle.

Game $\mathbf{G}_0$: This is the real-world attack game. In any game $\mathbf{G}_n$, we denote by S_n the event $\mathcal{V}(\mathsf{pk}, m, \sigma) = 1$, for a new signature σ.

Game $\mathbf{G}_1$: In this game, we make the classical simulation of the random oracles, with random answers for any new query, as shown on Figure 8. This game is clearly identical to the previous one. The $\mathcal{H}$ simulation may seem a bit intricate, but the bit c is never used. It will appear later.

$\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ oracles

Query $\mathcal{F}(w)$: if a record (w, t) appears in F-List, the answer is t.
Otherwise the answer t is chosen randomly: $t \in \{0,1\}^{k_0}$ and the record (w, t) is added in F-List.

Query $\mathcal{G}(w)$: if a record (w, g) appears in G-List, the answer is g.
Otherwise the answer g is chosen randomly: $g \in \{0,1\}^{k_1}$ and the record (w, g) is added in G-List.

Query $\mathcal{H}(m, r)$: one first sets $c = 0$ if the query is asked by the signing oracle, and $c = 1$ otherwise (by the adversary directly). If a record $(m, r, \star, \perp, w)$ appears in H-List:

►**Rule $\mathcal{H}$-Old$^{(1)}$**
The answer is w.

Otherwise the answer w is defined according to the following rule:

►**Rule $\mathcal{H}$-New$^{(1)}$**
Choose a random element $w \in \{0,1\}^{k_2}$. The record $(m, r, c, \perp, w)$ is added in H-List.

Note: the fourth component of the elements of this list will be explained later.

$\mathcal{S}$ oracle

For a sign-query $\mathcal{S}(m)$, one first chooses a random $r \in \{0,1\}^{k_1}$ and asks for $w = \mathcal{H}(m, r)$, $s = \mathcal{G}(w) \oplus r$ and $t = \mathcal{F}(w)$. Then one concatenates $y = 0\|w\|s\|t$ and computes the signature σ according to the following rule:

►**Rule $\mathcal{S}^{(1)}$**
Computes $\sigma = f^{-1}(y)$.

Fig. 8. Simulation of the Attack Game against PSS

Game $\mathbf{G}_2$: In this game, we introduce the random challenge $y^\star$, for which one is looking for $x^\star$ such that $y^\star = f(x^\star)$. Thus, we replace the random oracle $\mathcal{H}$ by the following simulation, which may abort:

►**Rule $\mathcal{H}$-New**$^{(2)}$

> Choose a random $u \in X$, then if $c = 0$, compute $z = y^\star \otimes f(u)$, otherwise compute $z = f(u)$, until the most significant bit of z is 0, but at most k_2 times (otherwise one aborts the game). Choose a random element $w \in \{0,1\}^{k_2}$. The record $(m, r, c, \perp, w)$ is added in H-List.

Let us remark that the number of calls to $\mathcal{H}$ is upper-bounded by $q_h + q_s$ (direct queries and queries asked by the signing oracle.) This game may only differ from the previous one during some $\mathcal{H}$-simulations, if the simulation aborts because z is still in the bad range, even after the k_2 attempts (event $\mathsf{BadRange}_2$). Using the Lemma 7, noting that

$$\Pr[\mathsf{S}_2 \mid \neg\mathsf{BadRange}_2] = \Pr[\mathsf{S}_1 \mid \neg\mathsf{BadRange}_2] \text{ and } \Pr[\mathsf{BadRange}_2] \leq \frac{q_h + q_s}{2^{k_2}},$$

one gets

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \frac{q_h + q_s}{2^{k_2}}. \tag{10}$$

Game $\mathbf{G}_3$: In the above game, one may have noted that z is uniformly distributed in X, because of the permutation property of f, with the conditioning that the most significant bit is 0. One can thus parse it into $0\|w\|s\|t$, where w is uniformly distributed in $\{0,1\}^{k_2}$:

►**Rule $\mathcal{H}$-New**$^{(3)}$

> Choose a random $u \in X$, then if $c = 0$, compute $z = y^\star \otimes f(u)$, otherwise compute $z = f(u)$, until the most significant bit of z is 0, but at most k_2 times (otherwise one aborts the game). Thereafter, z is parsed into $0\|w\|s\|t$, The record (m, r, c, u, w) is added in H-List.

This simulation is thus perfectly indistinguishable, since the additional field u in the H-List is never used. But note that $z = y^{\star c} \otimes f(u)$.

Game $\mathbf{G}_4$: Now, we furthermore anticipate some $\mathcal{F}$ or $\mathcal{G}$ answers, with random numbers, which is the case of the above s and t:

►**Rule $\mathcal{H}$-New**$^{(4)}$

> Choose a random $u \in X$, then if $c = 0$, compute $z = y^\star \otimes f(u)$, otherwise compute $z = f(u)$, until the most significant bit of z is 0, but at most k_2 times (otherwise one aborts the game). Thereafter, z is parsed into $0\|w\|s\|t$, and one adds the record (w, t) to the F-List and $(w, s \oplus r)$ to the G-List. The record (m, r, c, u, w) is added in H-List.

This game may only differ from the previous one if during some $\mathcal{H}$-simulations, $\mathcal{F}(w)$ or $\mathcal{G}(w)$ have already been defined (either by a direct query, or by a $\mathcal{H}$-simulation.)

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| \leq \frac{(q_h + q_s)(q_f + q_g + q_h + q_s)}{2^{k_2}}. \tag{11}$$

Game $\mathbf{G}_5$: Now, we simply abort if the signing oracle makes a $\mathcal{H}(m, r)$-query for some (m, r) that has already been asked to $\mathcal{H}$.

►**Rule $\mathcal{H}$-Old**$^{(5)}$

> If $c = 0$, then one aborts the game, otherwise the answer is w.

Because of the possible abortion

$$|\Pr[\mathsf{S}_5] - \Pr[\mathsf{S}_4]| \leq q_s(q_h + q_s)/2^{k_1}. \tag{12}$$

Game $\mathbf{G}_6$: In the last game, we replace the signing oracle by an easy simulation, returning the value u involved in the answer $\mathcal{H}(m, r)$, which defines $z = f(u)$:

►**Rule** $\mathcal{S}^{(6)}$
  Look up for (m, r, c, u, w) in H-List, and set $\sigma = u$.

The simulation is perfect since $c = 0$.

The event S_6 means that, at the end of that game, the adversary outputs a valid message/signature (m, σ). The latter satisfies: $y = f(\sigma) = b\|w\|s\|t$. Then one gets $r = s \oplus \mathcal{G}(w)$, and checks whether $b = 0$, $w = \mathcal{H}(m, r)$ and $t = f(w)$. Such a signature is valid

- without having queried $\mathcal{H}(m, r)$, which is possible with probability bounded by 2^{-k_2};
- with $y = y^\star \otimes f(u)$, where $(m, r, 1, u, w) \in$ H-List, and thus one gets $x^\star$.

$$\Pr[\mathsf{S}_6] \leq \mathsf{Succ}^{\mathsf{ow}}_f(t', k) + 2^{-k_2}, \tag{13}$$

where t' is the running time of the adversary, including the time for the simulations: $t' \leq t + (q_s + q_h) \cdot k_2 \cdot T_f$.

The important point in this security result is the very tight link between success probabilities, but also the almost linear time of the reduction. Thanks to this exact and efficient security result, RSA–PSS has become the new PKCS #1 v2.1 standard for signature [74]. Another variant has been proposed with message-recovery: PSS-R which allows one to include a large part of the message inside the signature. This makes a signed-message shorter than the size of the signature plus the size of the message, since the latter is inside the former one.

6 Public-Key Encryption

6.1 History

The RSA Encryption Scheme In the same paper [73] as the RSA signature scheme appeared, Rivest, Shamir and Adleman also proposed a public-key encryption scheme, thanks to the "trapdoor one-way permutation" property of the RSA function: the generation algorithm produces a large composite number $N = pq$, a public key e, and a private key d such that $e \cdot d = 1 \bmod \varphi(N)$. The encryption of a message m, encoded as an element in $\mathbb{Z}_N^\star$, is simply $c = m^e \bmod N$. This ciphertext can be easily decrypted thanks to the knowledge of d, $m = c^d \bmod N$. Clearly, this encryption is OW-CPA, relative to the RSA problem. The determinism makes a plaintext-checking oracle useless. Indeed, the encryption of a message m, under a public key pk is always the same, and thus it is easy to check whether a ciphertext c really encrypts m, by re-encrypting it. Therefore the RSA-encryption scheme is OW-PCA relative to the RSA problem as well.

Because of this determinism, it cannot be semantically secure: given the encryption c of either m_0 or m_1, the adversary simply computes $c' = m_0^e \bmod N$ and checks whether $c' = c$. Furthermore, with a small exponent e (*e.g.* $e = 3$), any security vanishes under a multi-user attack: given $c_1 = m^3 \bmod N_1$, $c_2 = m^3 \bmod N_2$ and $c_3 = m^3 \bmod N_3$, one can easily compute $m^3 \bmod N_1N_2N_3$ thanks to the Chinese Remainders Theorem, which is exactly m^3 in $\mathbb{Z}$ and therefore leads to an easy recovery of m.

The El Gamal Encryption Scheme In 1985, El Gamal [27] also designed a public-key encryption scheme based on the Diffie-Hellman key exchange protocol [25]: given a cyclic group $\mathcal{G}$ of order prime q and a generator $\mathbf{g}$, the generation algorithm produces a random element $x \in \mathbb{Z}_q^\star$ as private key, and a public key $\mathbf{y} = x \cdot \mathbf{g}$. The encryption of a message m, encoded as an element $\mathbf{m}$ in $\mathcal{G}$, is a pair $(\mathbf{c} = a \cdot \mathbf{g}, \mathbf{d} = a \cdot \mathbf{y} + \mathbf{m})$, for a random $a \in \mathbb{Z}_q$. This ciphertext can be easily decrypted thanks to the knowledge of x, since

$$a \cdot \mathbf{y} = ax \cdot \mathbf{g} = x \cdot \mathbf{c},$$

and thus $\mathbf{m} = \mathbf{d} - x \cdot \mathbf{c}$. This encryption scheme is well-known to be OW-CPA relative to the Computational Diffie-Hellman problem. It is also semantically secure (against chosen-plaintext attacks) relative to the Decisional Diffie-Hellman problem [85]. For OW-PCA, it relies on the Gap Diffie-Hellman problem [60].

As we have seen above, the expected security level is IND-CCA, whereas the RSA encryption just reaches OW-CPA under the RSA assumption, and the El Gamal encryption achieves IND-CPA under the **DDH** assumption. Can we achieve IND-CCA for practical encryption schemes?

6.2 A First Generic Construction

In [10], Bellare and Rogaway proposed the first generic construction which applies to any trapdoor one-way permutation f onto X. We need two hash functions $\mathcal{G}$ and $\mathcal{H}$:

$$\mathcal{G} : X \longrightarrow \{0,1\}^n \quad \text{and} \quad \mathcal{H} : \{0,1\}^\star \longrightarrow \{0,1\}^{k_1},$$

where n is the bit-length of the plaintexts, and k_1 a security parameter. Then the encryption scheme $\mathsf{BR} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be described as follows:

- $\mathcal{K}(1^k)$: specifies an instance of the function f, and of its inverse f^{-1}. The public key pk is therefore f and the private key sk is f^{-1}.
- $\mathcal{E}_{\mathsf{pk}}(m; r)$: given a message $m \in \{0,1\}^n$, and a random value $r \xleftarrow{R} X$, the encryption algorithm $\mathcal{E}_{\mathsf{pk}}$ computes

  $$a = f(r), \qquad b = m \oplus \mathcal{G}(r) \quad \text{and} \quad c = \mathcal{H}(m, r),$$

  and outputs the ciphertext $y = a \| b \| c$.
- $\mathcal{D}_{\mathsf{sk}}(a \| b \| c)$: thanks to the private key, the decryption algorithm $\mathcal{D}_{\mathsf{sk}}$ extracts

  $$r = f^{-1}(a), \quad \text{and next} \quad m = b \oplus \mathcal{G}(r).$$

  If $c = \mathcal{H}(m, r)$, the algorithm returns m, otherwise it returns "Reject."

About this construction, one can prove:

Theorem 10. *Let $\mathcal{A}$ be a* CCA*-adversary against the semantic security of the above encryption scheme* BR*. Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_d, q_g and q_h queries to the decryption oracle, and the hash functions $\mathcal{G}$ and $\mathcal{H}$, respectively. Then*

$$\mathsf{Succ}_f^{\mathsf{ow}}(\tau') \geq \frac{\varepsilon}{2} - \frac{2q_d}{2^{k_1}} - \frac{q_h}{2^n},$$
$$\text{with} \quad \tau' \leq \tau + (q_g + q_h) \cdot T_f,$$

where T_f denotes the time complexity for evaluating f.

Proof. In the following we use starred letters ($r^\star$, $a^\star$, $b^\star$, $c^\star$ and $y^\star$) to refer to the challenge ciphertext, whereas unstarred letters (r, a, b, c and y) refer to the ciphertext asked to the decryption oracle.

Game $\mathbf{G}_0$: A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary A_1 is fed with pk, the description of f, and outputs a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext $y^\star = a^\star \| b^\star \| c^\star$ of m_b. This ciphertext comes from a random $r^\star \stackrel{R}{\leftarrow} X$ and $a^\star = f(r^\star)$, $b^\star = m_b \oplus \mathcal{G}(r^\star)$ and $c^\star = \mathcal{H}(m_b, r^\star)$. On input $y^\star$, A_2 outputs bit b'. In both stages, the adversary is given additional access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$. The only requirement is that the challenge ciphertext $y^\star$ cannot be queried from the decryption oracle.

We denote by S_0 the event $b' = b$ and use a similar notation S_i in any $\mathbf{G}_i$ below. By definition, we have

$$\Pr[\mathsf{S}_0] = \frac{1}{2} + \frac{\varepsilon}{2}. \tag{14}$$

Game $\mathbf{G}_1$: In this game, one makes the classical simulation of the random oracles, with random answers for any new query, as shown on Figure 9. This game is clearly identical to the previous one.

$\mathcal{G}, \mathcal{H}$ Oracles

Query $\mathcal{G}(r)$: if a record (r, g) appears in G-List, the answer is g.
Otherwise the answer g is chosen randomly: $g \in \{0,1\}^n$ and the record (r, g) is added in G-List.

Query $\mathcal{H}(m, r)$: if a record (m, r, h) appears in H-List, the answer is h.
Otherwise the answer h is chosen randomly: $h \in \{0,1\}^{k_1}$ and the record (m, r, h) is added in H-List.

$\mathcal{D}$ Oracle

Query $\mathcal{D}_{\mathsf{sk}}(a\|b\|c)$: one applies the following rules:

►**Rule Decrypt−R**$^{(1)}$
Compute $r = f^{-1}(a)$;

Then, compute $m = b \oplus \mathcal{G}(r)$, and finally,

►**Rule Decrypt−H**$^{(1)}$
If $c = \mathcal{H}(m, r)$, one returns m, otherwise one returns "Reject."

Challenger

For two messages (m_0, m_1), flip a coin b and set $m^\star = m_b$.

►**Rule Chal−Hash**$^{(1)}$
Choose randomly $r^\star$, then set
$a^\star = f(r^\star)$,
$g^\star = \mathcal{G}(r^\star)$, $\quad b^\star = m^\star \oplus g^\star$,
$c^\star = \mathcal{H}(m^\star, r^\star)$.

Then, output $y^\star = a^\star \| b^\star \| c^\star$.

Fig. 9. Formal Simulation of the IND-CCA Game against the BR Construction

Game $\mathbf{G}_2$: In this game, one randomly chooses $h^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$, and uses it instead of $\mathcal{H}(m^\star, r^\star)$.

►**Rule Chal−Hash**$^{(2)}$
The value $h^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$ has been chosen ahead of time, choose randomly $r^\star$, then set $a^\star = f(r^\star)$, $g^\star = \mathcal{G}(r^\star)$, $b^\star = m^\star \oplus g^\star$, and $c^\star = h^+$.

The two games $\mathbf{G}_2$ and $\mathbf{G}_1$ are perfectly indistinguishable unless $(m^\star, r^\star)$ is asked for $\mathcal{H}$, either by the adversary or the decryption oracle. But the latter case is not possible, otherwise the decryption query would be the challenge ciphertext. More generally, we denote by AskR_2 the event that $r^\star$ has been asked to $\mathcal{G}$ or to $\mathcal{H}$, by the adversary. We have:

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{AskR}_2]. \tag{15}$$

Game $\mathbf{G}_3$: We start modifying the simulation of the decryption oracle, by rejecting any ciphertext $(a\|b\|c)$ for which the corresponding (m, r) has not been queried to $\mathcal{H}$:

▶**Rule Decrypt−H**$^{(3)}$
> Look up in H-List for (m, r, c). If such a triple does not exist, then output "Reject", otherwise output m.

Such a simulation differs from the previous one if the value c has been correctly guessed, by chance:

$$|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_2]| \leq \frac{q_d}{2^{k_1}} \qquad |\Pr[\mathsf{AskR}_3] - \Pr[\mathsf{AskR}_2]| \leq \frac{q_d}{2^{k_1}}. \tag{16}$$

Game $\mathbf{G}_4$: In this game, one randomly chooses $r^+ \xleftarrow{R} X$ and $g^+ \xleftarrow{R} \{0,1\}^n$, and uses r^+ instead of $r^\star$, as well as g^+ instead of $\mathcal{G}(r^\star)$.

▶**Rule Chal−Hash**$^{(4)}$
> The three values $r^+ \xleftarrow{R} X$, $g^+ \xleftarrow{R} \{0,1\}^n$ and $h^+ \xleftarrow{R} \{0,1\}^{k_1}$ have been chosen ahead of time, then set $a^\star = f(r^+), \quad b^\star = m^\star \oplus g^+, \quad c^\star = h^+$.

The two games $\mathbf{G}_4$ and $\mathbf{G}_3$ are perfectly indistinguishable unless $r^\star$ is asked for $\mathcal{G}$, either by the adversary or the decryption oracle. The former case has already been cancelled in the previous game, in AskR_3. The latter case does not make any difference since either $\mathcal{H}(m, r^\star)$ has been queried by the adversary, which falls in AskR_3, or the ciphertext is rejected in both games. We have:

$$\Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_3] \qquad \Pr[\mathsf{AskR}_4] = \Pr[\mathsf{AskR}_3]. \tag{17}$$

In this game, $m^\star$ is masked by g^+, a random value which never appears anywhere else. Thus, the input to $\mathcal{A}_2$ follows a distribution that does not depend on b. Accordingly:

$$\Pr[\mathsf{S}_4] = \frac{1}{2}. \tag{18}$$

Game $\mathbf{G}_5$: Finally, one randomly chooses $a^+ \xleftarrow{R} X$, which implicitly defines a random r^+ in X. Actually, a^+ is the given random challenge for which one is looking for the pre-image r^+.

▶**Rule Chal−Hash**$^{(5)}$
> The three values $a^+ \xleftarrow{R} X$, $g^+ \xleftarrow{R} \{0,1\}^n$ and $h^+ \xleftarrow{R} \{0,1\}^{k_1}$ have been chosen/given ahead of time, then set $a^\star = a^+, \quad b^\star = m^\star \oplus g^+, \quad c^\star = h^+$.

The two games $\mathbf{G}_5$ and $\mathbf{G}_4$ are perfectly indistinguishable, thanks to the permutation property of f.

Game $\mathbf{G}_6$: In the simulation of the decryption oracle, we may reject even earlier, if the corresponding r has not been queried to $\mathcal{G}$:

►**Rule Decrypt−R**$^{(6)}$

Look up in G-List for (r, g) such that $a = f(r)$. If no r is found, then output "Reject".

Such a simulation differs from the previous one if the value (m, r) has been queried to $\mathcal{H}$, while $\mathcal{G}(r)$ is unpredictable, and thus $m = \mathcal{G}(r) \oplus b$ is unpredictable too:

$$|\Pr[\mathsf{AskR}_6] - \Pr[\mathsf{AskR}_5]| \leq \frac{q_h}{2^n}. \tag{19}$$

One may now note that the event AskR_6 leads to the pre-image of a^+ by f in the queries asked to $\mathcal{G}$ and $\mathcal{H}$, by the adversary. By checking all of them, one gets it:

$$\Pr[\mathsf{AskR}_6] \leq \mathsf{Succ}^{\mathsf{ow}}_f(\tau + (q_g + q_h)T_f). \tag{20}$$

6.3 OAEP: the Optimal Asymmetric Encryption Padding.

Description The problem with the above generic construction is the high over-head. When one encrypts with a trapdoor one-way permutation onto X, one could hope the ciphertext to be an element in X, without anything else. In 1994, Bellare and Rogaway proposed such a more compact generic conversion [11], in the random-oracle model, the "Optimal Asymmetric Encryption Padding" (OAEP, see Figure 10), obtained from a trapdoor one-way permutation

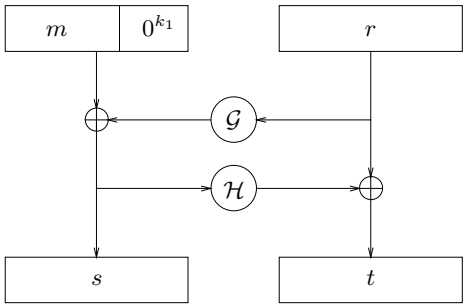

Fig. 10. Optimal Asymmetric Encryption Padding

f onto $\{0,1\}^k$, whose inverse is denoted by f^{-1}. We need two hash functions $\mathcal{G}$ and $\mathcal{H}$:

$$\mathcal{G} : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{k-k_0} \quad \text{and} \quad \mathcal{H} : \{0,1\}^{k-k_0} \longrightarrow \{0,1\}^{k_0},$$

for some k_0. We also need n and k_1 which satisfy $k = n + k_0 + k_1$. Then the encryption scheme $\mathsf{OAEP} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be described as follows:

- $\mathcal{K}(1^k)$: specifies an instance of the function f, and of its inverse f^{-1}. The public key pk is therefore f and the private key sk is f^{-1}.
- $\mathcal{E}_{\mathsf{pk}}(m; r)$: given a message $m \in \{0,1\}^n$, and a random value $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, the encryption algorithm $\mathcal{E}_{\mathsf{pk}}$ computes

  $$s = (m \| 0^{k_1}) \oplus \mathcal{G}(r) \quad \text{and} \quad t = r \oplus \mathcal{H}(s),$$

  and outputs the ciphertext $c = f(s, t)$.
- $\mathcal{D}_{\mathsf{sk}}(c)$: thanks to the private key, the decryption algorithm $\mathcal{D}_{\mathsf{sk}}$ extracts

  $$(s, t) = f^{-1}(c), \quad \text{and next} \quad r = t \oplus \mathcal{H}(s) \quad \text{and} \quad M = s \oplus \mathcal{G}(r).$$

  If $[M]_{k_1} = 0^{k_1}$, the algorithm returns $[M]^n$, otherwise it returns "Reject."

In the above description, $[M]_{k_1}$ denotes the k_1 least significant bits of M, while $[M]^n$ denotes the n most significant bits of M.

About the Security Paper [11] includes a proof that, provided f is a one-way trapdoor permutation, the resulting OAEP encryption scheme is both semantically secure and weakly plaintext-aware. This implies the semantic security against indifferent chosen-ciphertext attacks, also called security against lunchtime attacks (IND-CCA1). Indeed, the *Weak Plaintext-Awareness* means that the adversary cannot produce a new valid ciphertext, until it has seen any valid one, without knowing (awareness) the plaintext. This is more formally defined by the existence of a plaintext-extractor which, on input a ciphertext and the list of the query-answers of the random oracles, outputs the corresponding plaintext. This plaintext-extractor is thus enough for simulating the decryption oracle, but in the first step of the attack only. We briefly comment on the intuition behind (weak) plaintext-awareness. When the plaintext-extractor receives a ciphertext c, then:

- either s has been queried to $\mathcal{H}$ and r has been queried to $\mathcal{G}$, in which case the extractor finds the cleartext by inspecting the two query lists G-List and H-List,
- or else the decryption of (s, t) remains highly random and there is little chance to meet the redundancy 0^{k_1}: the plaintext extractor can safely declare the ciphertext invalid.

The argument collapses when the plaintext-extractor receives additional valid ciphertexts, since this puts additional implicit constraints on $\mathcal{G}$ and $\mathcal{H}$. These constraints cannot be seen by inspecting the query lists. Hence the requirement of a stronger notion of *plaintext-awareness*. In [7], Bellare, Desai, Rogaway and the author defined such a stronger notion which extends the previous *awareness* of the plaintext even after having seen valid ciphertexts. But such a plaintext-awareness notion had never been studied for OAEP, while it was still widely admitted.

Shoup's Counter-Example. In his papers [82, 83], Shoup showed that it was quite unlikely to extend the results of [11] to obtain adaptive chosen-ciphertext security, under the sole one-wayness of the permutation. His counter-example made use of the ad hoc notion of an *XOR-malleable* trapdoor one-way permutation: for such permutation f_0, one can compute $f_0(x \oplus a)$ from $f_0(x)$ and a, with non-negligible probability.

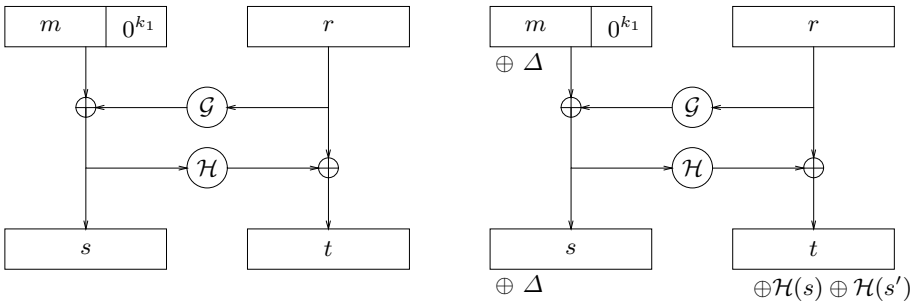

Fig. 11. Shoup's attack.

Let f_0 be such an XOR-malleable permutation. Define f by $f(s\|t) = s\|f_0(t)$. Clearly, f is also a trapdoor one-way permutation. However, it leads to a malleable encryption scheme as we now show. Start with a challenge ciphertext $y = f(s\|t) = s\|u$, where $s\|t$ is the output of the OAEP transformation on the redundant message $m\|0^{k_1}$ and the random string r (see Figure 11),

$$s = \mathcal{G}(r) \oplus (m\|0^{k_1}), \qquad t = \mathcal{H}(s) \oplus r \quad \text{and} \quad u = f_0(t).$$

Since f is the identity on its leftmost part, we know s, and can define $\Delta = \delta\|0^{k_1}$, for any random string δ, and $s' = s \oplus \Delta$. We then set $t' = \mathcal{H}(s') \oplus r = t \oplus (\mathcal{H}(s) \oplus \mathcal{H}(s'))$. The

XOR-malleability of f_0 allows one to obtain $u' = f_0(t')$ from $u = f_0(t)$ and $\mathcal{H}(s) \oplus \mathcal{H}(s')$, with significant probability. Finally, $y' = s' \| u'$ is a valid ciphertext of $m' = m \oplus \delta$, built from $r' = r$, since:

$$t' = f_0^{-1}(u') = t \oplus (\mathcal{H}(s) \oplus \mathcal{H}(s')) = \mathcal{H}(s') \oplus r, \qquad r' = \mathcal{H}(s') \oplus t' = r$$

and

$$s' \oplus \mathcal{G}(r') = \Delta \oplus s \oplus \mathcal{G}(r) = \Delta \oplus (m \| 0^{k_1}) = (m \oplus \delta) \| 0^{k_1}.$$

Note that the above definitely contradicts adaptive chosen-ciphertext security: asking the decryption of y' after having received the ciphertext y, an adversary obtains m' and easily recovers the actual cleartext m from m' and δ. Also note that Shoup's counter-example exactly stems from where the intuition developed at the end of the previous section failed: a valid ciphertext y' was created without querying the oracle at the corresponding random seed r', using in place the implicit constraint on $\mathcal{G}$ coming from the received valid ciphertext y.

Using methods from relativized complexity theory, Shoup [82, 83] built a non-standard model of computation, where there exists an XOR-malleable trapdoor one-way permutation. As a consequence, it is very unlikely that one can prove the IND-CCA security of the OAEP construction, under the sole one-wayness of the underlying permutation. Indeed, all methods of proof currently known still apply in relativized models of computation.

The Actual Security of OAEP Shoup [82, 83] furthermore provided a specific proof for RSA with public exponent 3. However, there is little hope of extending this proof for higher exponents. Hopefully, Fujisaki, Okamoto, Stern and the author provided a general security analysis, but under a stronger assumption about the underlying permutation [32, 33]. Indeed, we prove that the scheme is IND-CCA in the random-oracle model [10], relative to the *partial-domain* one-wayness of permutation f.

Partial-Domain One-Wayness. Let us first introduce this new computational assumption. Let f be a permutation $f : \{0,1\}^k \longrightarrow \{0,1\}^k$, which can also be written as

$$f : \{0,1\}^{n+k_1} \times \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n+k_1} \times \{0,1\}^{k_0},$$

with $k = n + k_0 + k_1$. In the original description of OAEP from [11], it is only required that f is a trapdoor one-way permutation. However, in the following, we consider two additional related problems, namely partial-domain one-wayness and set partial-domain one-wayness:

- Permutation f is (τ, ε)-one-way if any adversary $\mathcal{A}$ whose running time is bounded by τ has success probability $\mathsf{Succ}_f^{\mathsf{ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}_f^{\mathsf{ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = (s,t)].$$

- Permutation f is (τ, ε)-partial-domain one-way if any adversary $\mathcal{A}$ whose running time is bounded by τ has success probability $\mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = s].$$

- Permutation f is $(\ell, \tau, \varepsilon)$-set partial-domain one-way if any adversary $\mathcal{A}$, outputting a set of ℓ elements within time bound τ, has success probability $\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\mathcal{A})$ upper-bounded by ε, where

$$\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\mathcal{A}) = \Pr_{s,t}[s \in \mathcal{A}(f(s,t))].$$

We denote by $\mathsf{Succ}_f^{\mathsf{ow}}(\tau)$ (resp. $\mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\tau)$ and $\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\ell,\tau)$) the maximal success probability $\mathsf{Succ}_f^{\mathsf{ow}}(\mathcal{A})$ (resp. $\mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\mathcal{A})$ and $\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\mathcal{A})$). The maximum ranges over all adversaries whose running time is bounded by τ. In the third case, there is an obvious additional restriction on this range from the fact that $\mathcal{A}$ outputs sets with ℓ elements. It is clear that for any τ and $\ell \geq 1$,

$$\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\ell,\tau) \geq \mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\tau) \geq \mathsf{Succ}_f^{\mathsf{ow}}(\tau).$$

Note that, by randomly selecting an element in the set returned by an adversary to the set partial-domain one-wayness, one breaks partial-domain one-wayness with probability $\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\mathcal{A})/\ell$. This provides the following inequality

$$\mathsf{Succ}_f^{\mathsf{pd\text{-}ow}}(\tau) \geq \mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(\ell,\tau)/\ell.$$

However, for specific choices of f, more efficient reductions may exist. Also, in some cases, all three problems are polynomially equivalent. This is the case for the RSA permutation [73], hence the global security result for RSA-OAEP.

The Proof of Security In the following we use starred letters ($r^\star$, $s^\star$, $t^\star$ and $y^\star$) to refer to the challenge ciphertext, whereas unstarred letters (r, s, t and y) refer to the ciphertext asked to the decryption oracle.

The Intuition. Referring to our description of the intuition behind the original OAEP proof of security, given above, we can carry a more subtle analysis by distinguishing the case where s has not been queried from oracle $\mathcal{H}$ from the case where r has not been queried from $\mathcal{G}$. If s is not queried, then $\mathcal{H}(s)$ is random and uniformly distributed and r is necessarily defined as $t \oplus \mathcal{H}(s)$. This holds even if s matches with the string $s^\star$ coming from the valid ciphertext $y^\star$. There is a minute probability that $t \oplus \mathcal{H}(s)$ is queried from $\mathcal{G}$ or equals $r^\star$. Thus, $\mathcal{G}(r)$ is random: there is little chance that the redundancy 0^{k_1} is met and the extractor can safely reject.

We claim that r cannot match with $r^\star$, unless $s^\star$ is queried from $\mathcal{H}$. This is because $r^\star = t^\star \oplus \mathcal{H}(s^\star)$ equals $r = t \oplus \mathcal{H}(s)$ with minute probability. Thus, if r is not queried, then $\mathcal{G}(r)$ is random and we similarly infer that the extractor can safely reject. The argument fails only if $s^\star$ is queried.

Thus rejecting when it cannot combine elements of the lists G-List and H-List so as to build a pre-image of y, the plaintext-extractor is only wrong with minute probability, unless $s^\star$ has been queried by the adversary. This seems to show that OAEP leads to an IND-CCA encryption scheme if it is difficult to invert f "partially", which means: given $y^\star = f(s^\star \| t^\star)$, find $s^\star$.

The Strategy. Based on the intuition just described, we can formally prove that applying OAEP encoding to a trapdoor permutation which is difficult to partially invert, leads to an IND-CCA encryption scheme, hence the *partial-domain one-wayness*, which expresses the fact that the above partial inversion problem is difficult.

Chosen-ciphertext security is actually addressed, by turning the intuition explained above into a formal argument, involving a restricted variant of plaintext-awareness (where the list C of ciphertexts is limited to only one ciphertext, the challenge ciphertext $y^\star$).

Theorem 11. *Let $\mathcal{A}$ be a* CCA*-adversary against the semantic security of the encryption scheme* OAEP*. Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_d, q_g and q_h queries to the decryption oracle, and the hash functions $\mathcal{G}$ and $\mathcal{H}$, respectively. Then*

$$\mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(q_h, \tau') \geq \frac{\varepsilon}{2} - \left(\frac{2(q_d+2)(q_d+2q_g)}{2^{k_0}} + \frac{3q_d}{2^{k_1}} \right),$$
$$\text{with} \quad \tau' \leq \tau + q_g \cdot q_h \cdot (T_f + \mathcal{O}(1)),$$

where T_f denotes the time complexity for evaluating f.

The Plaintext-Extractor

Description. In order to prove the security against adaptive chosen-ciphertext attacks, it is necessary to simulate calls to a decryption oracle. As in the original paper [11], we design a plaintext-extractor (which is actually the same). But the analysis is more intricate because the success probability of the extractor cannot be estimated unconditionally but only relatively to some computational assumption. When the plaintext-extractor receives a ciphertext c, then:

- either s has been queried to $\mathcal{H}$ and r has been queried to $\mathcal{G}$, in which case the extractor finds the cleartext by inspecting the two query lists G-List and H-List. One indeed looks up for $(\gamma, \mathcal{G}_\gamma) \in$ G-List and $(\delta, \mathcal{H}_\delta) \in$ H-List. For such a pair, one defines $\sigma = \delta$, $\theta = \gamma \oplus \mathcal{H}_\delta$, $\mu = \mathcal{G}_\gamma \oplus \delta$, and checks whether $c = f(\sigma, \theta)$. If $[\mu]_{k_1} = 0^{k_1}$, then the tailing part is the plaintext.
- or else the decryption of (s, t) remains highly random and there is little chance to meet the redundancy 0^{k_1}: the plaintext extractor can safely declare the ciphertext invalid.

Comments. One can easily check that the output of the plaintext-extractor is uniquely defined, regardless of the ordering of the lists. To see this, observe that since f is a permutation, the value of $\sigma = s$ is uniquely defined and so is δ. Keep in mind that the G-List and H-List correspond to input-output pairs for the functions $\mathcal{G}$ and $\mathcal{H}$, and at most one output is related to a given input. This makes $\mathcal{H}_\delta$ uniquely defined as well. Similarly, $\theta = t$ is uniquely defined, and thus γ and $\mathcal{G}_\gamma$: at most one μ may be selected, which is output depending on whether $[\mu]_{k_1} = 0^{k_1}$ or not.

Furthermore, if both r and s have been queried by the adversary, the plaintext-extractor perfectly simulates the decryption oracle.

Proof In the following, $y^\star$ is the challenge ciphertext, obtained from the encryption oracle. Since we have in mind using the plaintext-extractor instead of the decryption oracle, trying to contradict semantic security, we assume that $y^\star$ is a ciphertext of m_b and denote by $r^\star$ its random seed. We have

$$r^\star = \mathcal{H}(s^\star) \oplus t^\star \quad \text{and} \quad \mathcal{G}(r^\star) = s^\star \oplus (m_b \| 0^{k_1}).$$

In what follows, all unstarred variables refer to the decryption queries.

We now present a proof with games which sequentially discard all cases for which the above plaintext-extractor may fail.

Game $\mathbf{G_0}$: A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary A_1 is fed with pk, the description of f, and outputs a pair of messages (m_0, m_1). Next a challenge ciphertext is

produced by flipping a coin b and producing a ciphertext $y^\star$ of m_b. This ciphertext comes from a random $r^\star \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and $s^\star$ and $t^\star$ such that $y^\star = f(s^\star, t^\star)$, where $s^\star = (m_b \| 0^{k_1}) \oplus \mathcal{G}(r^\star)$ and $t^\star = r^\star \oplus \mathcal{H}(s^\star)$. On input $y^\star$, A_2 outputs bit b'. In both stages, the adversary is given additional access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$. The only requirement is that the challenge ciphertext cannot be queried from the decryption oracle.

We denote by S_0 the event $b' = b$ and use a similar notation S_i in any $\mathbf{G}_i$ below. By definition, we have

$$\Pr[\mathsf{S}_0] = \frac{1}{2} + \frac{\varepsilon}{2}. \tag{21}$$

Game $\mathbf{G}_1$: In this game, one makes the classical simulation of the random oracles, with random answers for any new query, as shown on Figure 12. This game is clearly identical to the previous one.

$\mathcal{G}$, $\mathcal{H}$ Oracles

Query $\mathcal{G}(r)$: if a record (r, g) appears in G-List, the answer is g.
Otherwise the answer g is chosen randomly: $g \in \{0,1\}^{k-k_0}$ and the record (r, g) is added in G-List.

Query $\mathcal{H}(s)$: if a record (s, h) appears in H-List, the answer is h.
Otherwise the answer h is chosen randomly: $h \in \{0,1\}^{k_0}$ and the record (s, h) is added in H-List.

$\mathcal{D}$ Oracle

Query $\mathcal{D}_{\mathsf{sk}}(c)$:the value M is defined according to the following rules:

►**Rule Decrypt−Init**$^{(1)}$
Compute $(s, t) = f^{-1}(c)$;

Look up for $(s, h) \in$ H-List:

– if the record is found, compute $r = t \oplus h$.
Look up for $(r, g) \in$ G-List:
• if the record is found
►**Rule Decrypt−SR**$^{(1)}$
$h = \mathcal{H}(s)$, $r = t \oplus h$,
$g = \mathcal{G}(r)$, $M = s \oplus g$.
• otherwise
►**Rule Decrypt−SnoR**$^{(1)}$
same as rule **Decrypt−SR**$^{(1)}$.
– otherwise
►**Rule Decrypt−noS**$^{(1)}$
same as rule **Decrypt−SR**$^{(1)}$.

If $[M]_{k_1} = 0^{k_1}$, one returns $m = [M]^n$, otherwise one returns "Reject."

Challenger

For two messages (m_0, m_1), flip a coin b and set $m^\star = m_b$, $M^\star = m^\star \| 0^{k_1}$.

►**Rule Chal−Hash**$^{(1)}$
Choose randomly $r^\star$, then set
$g^\star = \mathcal{G}(r^\star)$, $s^\star = M^\star \oplus g^\star$,
$h^\star = \mathcal{H}(s^\star)$, $t^\star = r^\star \oplus h^\star$.

►**Rule Chal−Output**$^{(1)}$
Compute and output $y^\star = f(s^\star, t^\star)$.

Fig. 12. Formal Simulation of the IND-CCA Game against OAEP

Game $\mathbf{G}_2$: In this game, one randomly chooses $r^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and $g^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$, and uses r^+ instead of $r^\star$, as well as g^+ instead of $\mathcal{G}(r^\star)$.

►**Rule Chal−Hash**$^{(2)}$

The two values $r^+ \xleftarrow{R} \{0,1\}^{k_0}$, $g^+ \xleftarrow{R} \{0,1\}^{k-k_0}$ have been chosen ahead of time, then set $r^\star = r^+, \quad g^\star = g^+,$
$s^\star = M^\star \oplus g^+, \quad h^\star = \mathcal{H}(s^\star), \quad t^\star = r^+ \oplus h^\star.$

The two games $\mathbf{G}_2$ and $\mathbf{G}_1$ are perfectly indistinguishable unless $r^\star$ is asked for $\mathcal{G}$, either by the adversary or by the decryption oracle. We define this event AskG_2. We have:

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{AskG}_2]. \tag{22}$$

In this game, g^+ is used in (s,t) but does not appear in the computation since $\mathcal{G}(r^+)$ is not defined to be equal to g^+. Thus, the input to $\mathcal{A}_2$ follows a distribution that does not depend on b. Accordingly:

$$\Pr[\mathsf{S}_2] = \frac{1}{2}. \tag{23}$$

Game $\mathbf{G}_3$: We start dealing with the decryption oracle, which has remained perfect up to this game, but using the ability to invert f. We first make the decryption oracle reject all ciphertexts c such that the corresponding r value has not been previously queried from $\mathcal{G}$ by the adversary.

►**Rule Decrypt−SnoR**$^{(3)}$

$g = \mathcal{G}(r), \quad M = 1^k.$

This new rule leads to a Reject since the 0^{k_1} is not verified. This makes a difference only if c is a valid ciphertext, while $\mathcal{G}(r)$ has not been asked. Since $\mathcal{G}(r)$ is uniformly distributed, equality $[s \oplus \mathcal{G}(r)]_{k_1} = 0^{k_1}$ happens with probability $1/2^{k_1}$. Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskG}_3] - \Pr[\mathsf{AskG}_2]| \leq \frac{q_d}{2^{k_1}}. \tag{24}$$

Note that we cannot remove the query $\mathcal{G}(r)$ from this rule, even if it would not change anything in the simulation of the output of this decryption. However, it would remove a pair (r,g) from $\mathsf{G\text{-}List}$, which could be $r^\star$ itself, and this would have a non-negligible impact on the event AskG_3.

Game $\mathbf{G}_4$: We now make the decryption oracle reject all ciphertexts c such that the corresponding s value has not been previously queried from $\mathcal{H}$ by the adversary.

►**Rule Decrypt−noS**$^{(4)}$

$h = \mathcal{H}(s), \quad r = t \oplus h,$
$g = \mathcal{G}(r), \quad M = 1^k.$

This makes a difference only if y is a valid ciphertext, while $\mathcal{H}(s)$ has not been asked. First, since $r = \mathcal{H}(s) \oplus t$ is uniformly distributed, it has been queried from $\mathcal{G}$ with probability less than $(q_g + q_d)/2^{k_0}$. Then, if $\mathcal{G}(r)$ has not been queried, the redundancy is satisfied with probability less than $1/2^{k_1}$. Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskG}_4] - \Pr[\mathsf{AskG}_3]| \leq \frac{q_d(q_g + q_d)}{2^{k_0}} + \frac{q_d}{2^{k_1}}. \tag{25}$$

Game $\mathbf{G}_5$: Here, we can make the first formal modification in the previous game since, whatever the h-value is, the message M is 1^k, and g and h are never revealed:

►**Rule Decrypt−noS**$^{(5)}$
$\quad h = \mathcal{H}(s), \quad M = 1^k.$

This will just postpone the definition of $\mathcal{G}(r)$ and also remove one pair (r, g) from G-List. The latter removal may have some impact:

- on the simulation of a later decryption c', if $r' = r$ was found in the previous game, but that is no longer in the list. A rule **Decrypt−SR** is thus replaced by the rule **Decrypt−SnoR**, which means that $g' = g$ was just defined in the modified rule, and never revealed (by any means: no information is leaked.) Therefore, the probability for M' to satisfy the redundancy was 2^{-k_1};
- the removed r could be $r^\star$, but this is $t \oplus \mathcal{H}(s)$, for $s \notin$ H-List. Such a case is bounded by 2^{-k_0}.

Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskG}_5] - \Pr[\mathsf{AskG}_4]| \leq q_d \times \left(\frac{1}{2^{k_0}} + \frac{1}{2^{k_1}}\right). \tag{26}$$

Game G$_6$: We follow in making formal modifications:

►**Rule Decrypt−noS**$^{(6)}$
$\quad M = 1^k.$

This will postpone the definition of $\mathcal{H}(s)$, and also remove the pair (s, h) from H-List. The latter removal may have some impact on the simulation of a later decryption c': if $s' = s$ was found in the previous game, but that is no longer in the list:

- a rule **Decrypt−SnoR** is replaced by the rule **Decrypt−noS** (which just cancels r' from G-List), which means that $h' = h$ was just defined in the modified rule, and never revealed. The probability for r' to be equal to $r^\star$ is 2^{-k_0}.
- a rule **Decrypt−SR** is replaced by the rule **Decrypt−noS**, which means that $h' = h$ was just defined in the modified rule, and never revealed. The probability for $r' = t' \oplus h'$ to be in G-List was less than $q_g/2^{k_0}$, which is an upper-bound of this case to appear.

In both cases, the decryption is anyway still the same. Summing up for all decryption queries, we get

$$|\Pr[\mathsf{AskG}_6] - \Pr[\mathsf{AskG}_5]| \leq \frac{q_d(q_g + 1)}{2^{k_0}}. \tag{27}$$

Furthermore, in the decryption simulation, when both r and s have been asked, no new query occurs:

►**Rule Decrypt−SR**$^{(6)}$
$\quad M = s \oplus g.$

As a consequence, the new decryption simulation makes no new $\mathcal{H}$-query.

Game G$_7$: We now define $s^\star$ independently of anything else, as well as $\mathcal{H}(s^\star)$, by randomly choosing $s^+ \xleftarrow{R} \{0,1\}^{k-k_0}$ and $h^+ \xleftarrow{R} \{0,1\}^{k_0}$, and using s^+ instead of $s^\star$, as well as h^+ instead of $\mathcal{H}(s^\star)$. The only change is that $s^\star = s^+$ instead of $M^\star \oplus g^+$, which in some sense defines $g^+ = M^\star \oplus s^+$ but we do not need it. The game obeys the following rule:

►**Rule Chal−Hash**$^{(7)}$

The three values $r^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, $s^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$ and $h^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ have been chosen ahead of time, then set $s^\star = s^+, \quad t^\star = r^+ \oplus h^+$.

The two games $\mathbf{G}_7$ and $\mathbf{G}_6$ are perfectly indistinguishable unless $s^\star$ is asked for $\mathcal{H}$ by the adversary, or used by the decryption oracle. The former event is denoted AskH_7, while the latter makes a difference only if the rule **Decrypt−SR**$^{(6)}$ was used, with an accepted ciphertext, or the rule **Decrypt−SnoR**$^{(6)}$ was used, with $r = r^\star$ (because this rule becomes **Decrypt−noS**$^{(6)}$, where no $\mathcal{G}(r)$ query is done, since it could have been $r^\star$, and thus made the event AskG happen.)

We thus insist here on that the event AskH_7 denotes the fact that $s^\star$ is asked for $\mathcal{H}$ by the adversary, whereas the event AskG denotes the fact that $r^\star$ is asks for $\mathcal{G}$ by the adversary or the decryption oracle/simulation.

Let us briefly deal with the bad cases:

- the rule **Decrypt−SR**$^{(6)}$ was used, with an accepted ciphertext. This means that there exists a valid ciphertext $c = f(s^\star \| t)$ that is queried to the decryption oracle, with the corresponding r queried to $\mathcal{G}$, where $r = t \oplus \mathcal{H}(s^\star) = t \oplus t^\star \oplus r^+$, and r^+ is a random value.
- the rule **Decrypt−SnoR**$^{(6)}$ was used, with $r = r^+$, where r^+ is a random value.

$$|\Pr[\mathsf{AskG}_7] - \Pr[\mathsf{AskG}_6]| \leq \Pr[\mathsf{AskH}_7] + \frac{q_d(q_g + q_d)}{2^{k_0}} + \frac{q_d}{2^{k_0}}. \tag{28}$$

In this new game, $r^\star = t^\star \oplus h^+$ is uniformly distributed, and independent of the adversary's view, since h^+ is never revealed:

$$\Pr[\mathsf{AskG}_7] \leq \frac{q_g + q_d}{2^{k_0}}, \tag{29}$$

where q_g and q_d denote the number of queries asked by the adversary to $\mathcal{G}$, or to the decryption oracle, respectively. As a consequence,

$$\Pr[\mathsf{AskG}_2] \leq \frac{3q_d}{2^{k_1}} + \frac{(2q_d+1)(q_g+q_d)}{2^{k_0}} + \frac{q_d(q_g+3)}{2^{k_0}} + \Pr[\mathsf{AskH}_7] \tag{30}$$

Game $\mathbf{G}_8$: Finally, we define $s^\star$ and $t^\star$ independently of anything else, by randomly choosing $s^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$ and $t^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$:

►**Rule Chal−Hash**$^{(8)}$

The two values $s^+ \stackrel{R}{\leftarrow} \{0,1\}^{k-k_0}$ and $t^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ have been chosen ahead of time, then set $s^\star = s^+, \quad t^\star = t^+$.

The two games $\mathbf{G}_8$ and $\mathbf{G}_7$ are perfectly indistinguishable.

Game $\mathbf{G}_9$: We now completely manufacture the challenge ciphertext: we randomly choose $y^+ \stackrel{R}{\leftarrow} \{0,1\}^k$, and simply set $y^\star = y^+$, ignoring the encryption algorithm altogether. This implicitly defines s^+ and t^+, because of the permutation property of f. Actually, y^+ is the given random challenge for which one is looking for the partial pre-image s^+.

►**Rule Chal−Hash**$^{(9)}$

Do nothing.

►**Rule Chal−Output**$^{(9)}$

The challenge $y^+ \stackrel{R}{\leftarrow} \{0,1\}^k$ has been given ahead of time, then set and output $y^\star = y^+$.

The distribution of $y^\star$ remains the same: due to the fact that f is a permutation, the previous method defining $y^\star = f(s^\star \| t^\star)$, with $s^\star = s^+$ and $t^\star = t^+$ was already generating a uniform distribution over the k-bit elements.

Game $\mathbf{G}_{10}$: Before concluding, one may remark that the new simulation of the decryption oracle is exactly the way the plaintext-extractor previously explained would operate, with some extra but unuseful $\mathcal{G}$-queries. Since we do not care anymore about the event AskG_{10}, they can be simplified:

▶**Rule Decrypt−SR**$^{(10)}$
  $M = s \oplus g$.

▶**Rule Decrypt−SnoR**$^{(10)}$
  $M = 1^k$.

▶**Rule Decrypt−noS**$^{(10)}$
  $M = 1^k$.

Finally, simply outputting the list of queries to $\mathcal{H}$ during this game, one gets

$$\Pr[\mathsf{AskH}_{10}] \leq \mathsf{Succ}_f^{\mathsf{s\text{-}pd\text{-}ow}}(q_h, \tau'). \tag{31}$$

To conclude the proof of Theorem 11, one just has to comment on the running time τ'. Although the plaintext-extractor is called q_d times, there is no q_d multiplicative factor in the bound for τ'. This comes from a simple bookkeeping argument. Instead of only storing the lists $\mathsf{G\text{-}List}$ and $\mathsf{H\text{-}List}$, one stores an additional structure consisting of tuples $(\gamma, \mathcal{G}_\gamma, \delta, \mathcal{H}_\delta, y)$. A tuple is included only for $(\gamma, \mathcal{G}_\gamma) \in \mathsf{G\text{-}List}$ and $(\delta, \mathcal{H}_\delta) \in \mathsf{H\text{-}List}$. For such a pair, one defines $\sigma = \delta$, $\theta = \gamma \oplus \mathcal{H}_\delta$, $\mu = \mathcal{G}_\gamma \oplus \delta$, and computes $y = f(\sigma, \theta)$. If $[\mu]_{k_1} = 0^{k_1}$, one stores the tuple $(\gamma, \mathcal{G}_\gamma, \delta, \mathcal{H}_\delta, y)$. The cumulative cost of maintaining the additional structure is $q_g \cdot q_h \cdot (T_f + \mathcal{O}(1))$ but, handling it to the plaintext-extractor allows one to output the expected decryption of y, by table lookup, in constant time. Of course, a time-space tradeoff is possible, giving up the additional table, but raising the computing time to $q_d \cdot q_g \cdot q_h \cdot (T_f + \mathcal{O}(1))$. □

Particular Case: RSA–OAEP Theorem 11 unfortunately requires a very strong assumption on the trapdoor permutation: the partial-domain one-wayness. Hopefully, in [33], we furthermore proved that for RSA, this is not a stronger assumption than the classical RSA assumption:

Lemma 12. *Let $\mathcal{A}$ be an algorithm that outputs a q-set containing $k-k_0$ of the most significant bits of the e-th root of its input (partial-domain RSA, for any modulus N, which $2^{k-1} < N < 2^k$ and $k > 2k_0$), within time bound t, with probability ε. There exists an algorithm that solves the RSA problem (N, e) with success probability ε', within time bound t' where*

$$\varepsilon' \geq \varepsilon \times (\varepsilon - 2^{2k_0 - k + 6}), \qquad t' \leq 2t + q^2 \times \mathcal{O}(k^3).$$

Combining this lemma with the previous general security result about OAEP, one gets

Theorem 13. *Let $\mathcal{A}$ be a* CCA–adversary against the "semantic security" of RSA–OAEP (where the modulus is k-bit long, $k > 2k_0$), with running time bounded by t and advantage ε, making q_d, q_g and q_h queries to the decryption oracle, and the hash functions G and H, respectively. Then the RSA problem can be solved with probability ε' greater than

$$\frac{\varepsilon^2}{4} - \varepsilon \cdot \left(\frac{2(q_d + 2)(q_d + 2q_g)}{2^{k_0}} + \frac{3q_d}{2^{k_1}} + \frac{32}{2^{k-2k_0}} \right)$$

within time bound $t' \leq 2t + q_h \cdot (q_h + 2q_g) \times \mathcal{O}(k^3)$.

There is actually a slight inconsistency in piecing together the two above results, coming from the fact that RSA is not a permutation over k-bit strings. Research papers usually ignore the problem. Of course, standards have to cope with it. Observe that one may decide only to encode a message of $n - 8$ bits, where n is $k - k_0 - k_1$ as before, as is done in the PKCS #1 standard. The additional redundancy leading bit can be treated the same way as the 0^{k_1} redundancy, especially with respect to decryption. However, this is not enough since $G(r)$ might still carry the string $(s\|t)$ outside the domain of the RSA encryption function. An easy way out is to start with another random seed if this happens. On average, 256 trials will be enough.

This security result does not achieve the practical security, because of the expensive reduction. In [33], we improved the reduction cost, with a more intricate proof. More precisely:

Theorem 14. *Let $\mathcal{A}$ be a* CCA*–adversary against the "semantic security" of RSA–OAEP (where the modulus is k-bit long, $k > 2k_0$), with running time bounded by t and advantage ε, making q_d, q_g and q_h queries to the decryption oracle, and the hash functions G and H, respectively. Then the RSA problem can be solved with probability ε' greater than*

$$\varepsilon^2 - 2\varepsilon \cdot \left(\frac{2q_d q_g + q_d + q_g}{2^{k_0}} + \frac{2q_d}{2^{k_1}} + \frac{32}{2^{k-2k_0}} \right)$$

within time bound $t' \leq 2t + q_h \cdot (q_h + 2q_g) \times \mathcal{O}(k^3)$.

Unfortunately, the reduction is still very expensive, and is thus meaningful for huge moduli only, more than 4096-bit long. Indeed, the RSA inverter we can build, thanks to this reduction, has a complexity at least greater than $q_h \cdot (q_h + 2q_g) \times \mathcal{O}(k^3)$. As already remarked, the adversary can ask up to 2^{60} queries to the hash functions, and thus this overhead in the inversion is at least 2^{151}. However, current factoring algorithms can factor up to 4096 bit-long integers within this number of basic operations (see [47] for complexity estimates of the most efficient factoring algorithms).

Anyway, the formal proof shows that the global design of OAEP is sound, and that it is still probably safe to use it in practice (*e.g.* in PKCS #1 v2.0, while being very careful during the implementation [49]).

6.4 REACT: a Rapid Enhanced-security Asymmetric Cryptosystem Transform

Unfortunately, there is no hope to use OAEP with any **DL**-based primitive, because of the "permutation" requirement. The OAEP construction indeed requires the primitive to be a permutation (trapdoor partial-domain one-way), which is the case of the RSA function. However, the only trapdoor problem known in the **DL**-setting is the Diffie-Hellman problem, and it does not provide any bijection. Thus, first Fujisaki and Okamoto [30] proposed a generic conversion from any IND-CPA scheme into an IND-CCA one, in the random-oracle model. While applying this conversion to the above El Gamal encryption (see Section 6.1), one obtains an IND-CCA encryption scheme relative to the **DDH** problem. Later, independently, Fujisaki and Okamoto [31] and the author [62] proposed better generic conversions since they apply to any OW-CPA scheme to make it into an IND-CCA one, still in the random-oracle model.

This high security level is just at the cost of two more hashings for the new encryption algorithm, as well as two more hashings but one re-encryption for the new decryption process.

Description The re-encryption cost is the main drawback of these conversions for practical purposes. Therefore, Okamoto and the author tried and succeeded in providing a conversion

that is both secure and efficient [59]: REACT, for "Rapid Enhanced-security Asymmetric Cryptosystem Transform". It is actually quite similar to the BR construction, excepted that it applies to any trapdoor one-way function, not permutations only.

$\mathcal{K}'$: **Key Generation** $\rightarrow (\mathsf{pk}, \mathsf{sk})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^\kappa)$ $\rightarrow (\mathsf{pk}, \mathsf{sk})$
$\mathcal{E}'$: **Encryption of** $m \in \mathbf{M}' = \{0,1\}^\ell \rightarrow (a, b, c)$
$R \in \mathbf{M}$ and $r \in \mathbf{R}$ are randomly chosen $a = \mathcal{E}_{\mathsf{pk}}(R; r) \quad b = m \oplus \mathcal{G}(R) \quad c = \mathcal{H}(R, m, a, b)$ $\rightarrow (a, b, c)$ is the ciphertext
$\mathcal{D}'$: **Decryption of** (a, b, c)
Given $a \in \mathbf{C}$, $b \in \{0,1\}^\ell$ and $c \in \{0,1\}^\kappa$ $R = \mathcal{D}_{\mathsf{sk}}(a) \qquad m = b \oplus \mathcal{G}(R)$ if $c = \mathcal{H}(R, m, a, b)$ and $R \in \mathbf{M} \rightarrow m$ is the plaintext (otherwise, "Reject: invalid ciphertext")

Fig. 13. Rapid Enhanced-security Asymmetric Cryptosystem Transform $\mathsf{REACT} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$

The latter conversion is indeed very efficient in many senses

- the computational overhead is just the cost of two hashings for *both* encryption and decryption
- if one can break IND-CCA of the resulting scheme with an expected time T, one can break OW-PCA of the basic scheme within almost the same amount of time, with a low overhead (not as with OAEP). It thus provides a *practical* security result.

Let us describe this generic conversion REACT [59] on any encryption scheme $\mathsf{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

$$\mathcal{E} : \mathbf{PK} \times \mathbf{M} \times \mathbf{R} \rightarrow \mathbf{C}, \qquad \mathcal{D} : \mathbf{SK} \times \mathbf{C} \rightarrow \mathbf{M},$$

where **PK** and **SK** are the sets of the public and private keys, **M** is the messages space, **C** is the ciphertexts space and **R** is the random coins space. One should remark that **R** may be small and even empty, with a deterministic encryption scheme, such as RSA. But in many other cases, such as the El Gamal encryption, it is as large as **M**. We also need two hash functions $\mathcal{G}$ and $\mathcal{H}$,

$$\mathcal{G} : \mathbf{M} \rightarrow \{0,1\}^\ell, \qquad \mathcal{H} : \mathbf{M} \times \{0,1\}^\ell \times \mathbf{C} \times \{0,1\}^\ell \rightarrow \{0,1\}^\kappa,$$

where κ is the security parameter, while ℓ denotes the size of the messages to encrypt. The REACT conversion is depicted on Figure 13.

Security Result About this construction, one can prove:

Theorem 15. *Let $\mathcal{A}$ be a* CCA*-adversary against the semantic security of the encryption scheme* $\mathsf{REACT} = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$. *Assume that $\mathcal{A}$ has advantage ε and running time τ and makes q_d, q_g and q_h queries to the decryption oracle, and the hash functions $\mathcal{G}$ and $\mathcal{H}$, respectively. Then*

$$\mathsf{Succ}_{\mathsf{S}}^{\mathsf{ow-pca}}(\tau') \geq \frac{\varepsilon}{2} - \frac{2q_d}{2^\kappa} - \frac{q_h}{2^\ell},$$
$$\text{with} \quad \tau' \leq \tau + (q_g + q_h) \cdot T_{\mathsf{pca}},$$

where T_{pca} denotes the times required by the PCA *oracle to answer any query.*

Proof. In the following we use starred letters ($r^\star$, $a^\star$, $b^\star$, $c^\star$ and $y^\star$) to refer to the challenge ciphertext, whereas unstarred letters (r, a, b, c and y) refer to the ciphertext asked to the decryption oracle.

Game $\mathbf{G}_0$: A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary A_1 is fed with pk, and outputs a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext $y^\star = a^\star \| b^\star \| c^\star$ of m_b. This ciphertext comes from random $R^\star \stackrel{R}{\leftarrow} \mathbf{M}$ and $r^\star \stackrel{R}{\leftarrow} \mathbf{R}$ and $a^\star = \mathcal{E}_{\mathsf{pk}}(R^\star, r^\star)$, $b^\star = m_b \oplus \mathcal{G}(R^\star)$ and $c^\star = \mathcal{H}(R^\star, m_b, a^\star, b^\star)$. On input $y^\star$, A_2 outputs bit b'. In both stages, the adversary is given additional access to the decryption oracle $\mathcal{D}'_{\mathsf{sk}}$. The only requirement is that the challenge ciphertext cannot be queried from the decryption oracle.

We denote by S_0 the event $b' = b$ and use a similar notation S_i in any $\mathbf{G}_i$ below. By definition, we have

$$\Pr[\mathsf{S}_0] = \frac{1}{2} + \frac{\varepsilon}{2}. \tag{32}$$

Game $\mathbf{G}_1$: In this game, one makes the classical simulation of the random oracles, with random answers for any new query, as shown on Figure 14. This game is clearly identical to the previous one.

$\mathcal{G}, \mathcal{H}$ Oracles

Query $\mathcal{G}(r)$: if a record (r, g) appears in G-List, the answer is g.
Otherwise the answer g is chosen randomly: $g \in \{0,1\}^\ell$ and the record (r, g) is added in G-List.

Query $\mathcal{H}(R, m, a, b)$: if a record (R, m, a, b, h) appears in H-List, the answer is h.
Otherwise the answer h is chosen randomly: $h \in \{0,1\}^\kappa$ and the record (R, m, a, b, h) is added in H-List.

$\mathcal{D}'$ Oracle

Query $\mathcal{D}'_{\mathsf{sk}}(a\|b\|c)$: one applies the following rules:

►**Rule Decrypt−R**$^{(1)}$
Compute $R = \mathcal{D}_{\mathsf{sk}}(a)$;

Then, compute $m = b \oplus \mathcal{G}(R)$, and finally,

►**Rule Decrypt−H**$^{(1)}$
If $c = \mathcal{H}(R, m, a, b)$, one returns m, otherwise one returns "Reject."

Challenger

For two messages (m_0, m_1), flip a coin b and set $m^\star = m_b$.

►**Rule Chal−Hash**$^{(1)}$
Choose randomly $R^\star$ and $r^\star$, then set
$a^\star = \mathcal{E}_{\mathsf{pk}}(R^\star, r^\star)$,
$g^\star = \mathcal{G}(R^\star)$, $\quad b^\star = m^\star \oplus g^\star$,
$c^\star = \mathcal{H}(R^\star, m^\star, a^\star, b^\star)$.

Then, output $y^\star = a^\star \| b^\star \| c^\star$.

Fig. 14. Formal Simulation of the IND-CCA Game against REACT

Game $\mathbf{G}_2$: In this game, one randomly chooses $h^+ \stackrel{R}{\leftarrow} \{0,1\}^\kappa$, and uses it instead of $\mathcal{H}(R^\star, m^\star, a^\star, b^\star)$.

►**Rule Chal−Hash**$^{(2)}$
The value $h^+ \stackrel{R}{\leftarrow} \{0,1\}^\kappa$ has been chosen ahead of time, choose randomly $R^\star$ and $r^\star$, then set
$a^\star = \mathcal{E}_{\mathsf{pk}}(R^\star, r^\star)$, $\quad g^\star = \mathcal{G}(R^\star)$, $\quad b^\star = m^\star \oplus g^\star$, $\quad c^\star = h^+$.

The two games $\mathbf{G}_2$ and $\mathbf{G}_1$ are perfectly indistinguishable unless $(R^\star, m^\star, a^\star, b^\star)$ is asked for $\mathcal{H}$, either by the adversary or the decryption oracle. But the latter case is not possible, otherwise the decryption query would be the challenge ciphertext itself. More generally, we denote by AskR_2 the event that $R^\star$ has been asked to $\mathcal{G}$ or to $\mathcal{H}$, by the adversary. We have:

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{AskR}_2] \tag{33}$$

Game $\mathbf{G}_3$: We start modifying the simulation of the decryption oracle, by rejecting any ciphertext $(a\|b\|c)$ for which the corresponding (R, m, a, b) has not been queried to $\mathcal{H}$:

►**Rule Decrypt−H**$^{(3)}$
> Look up in $\mathsf{H\text{-}List}$ for (R, m, a, b, c). If such a triple does not exist, then output "Reject", otherwise output m.

Such a simulation differs from the previous one if the value c has been correctly guessed, by chance:

$$|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_2]| \leq \frac{q_d}{2^\kappa} \qquad |\Pr[\mathsf{AskR}_3] - \Pr[\mathsf{AskR}_2]| \leq \frac{q_d}{2^\kappa}. \tag{34}$$

Game $\mathbf{G}_4$: In this game, one randomly chooses $R^+ \xleftarrow{R} \mathbf{M}$ and $r^+ \xleftarrow{R} \mathbf{R}$, and $g^+ \xleftarrow{R} \{0,1\}^\ell$, and uses R^+ instead of $R^\star$, r^+ instead of $r^\star$, as well as g^+ instead of $\mathcal{G}(R^\star)$.

►**Rule Chal−Hash**$^{(4)}$
> The four values $R^+ \xleftarrow{R} \mathbf{M}$, $r^+ \xleftarrow{R} \mathbf{R}$, $g^+ \xleftarrow{R} \{0,1\}^\ell$ and $h^+ \xleftarrow{R} \{0,1\}^\kappa$ have been chosen ahead of time, then set $a^\star = \mathcal{E}_{\mathsf{pk}}(R^+, r^+)$, $\quad b^\star = m^\star \oplus g^+$, $\quad c^\star = h^+$.

The two games $\mathbf{G}_4$ and $\mathbf{G}_3$ are perfectly indistinguishable unless $R^\star$ is asked for $\mathcal{G}$, either by the adversary or the decryption oracle. The former case has already been cancelled in the previous game in AskR_3. The latter case makes no difference since either $\mathcal{H}(R^\star, m, a, b)$ has been queried by the adversary, which falls in AskR_3, or the ciphertext is rejected in both games. We have:

$$\Pr[\mathsf{S}_4] = \Pr[\mathsf{S}_3] \qquad \Pr[\mathsf{AskR}_4] = \Pr[\mathsf{AskR}_3]. \tag{35}$$

In this game, $m^\star$ is masked by g^+, a random value which never appears anywhere else. Thus, the input to $\mathcal{A}_2$ follows a distribution that does not depend on b. Accordingly:

$$\Pr[\mathsf{S}_4] = \frac{1}{2}. \tag{36}$$

Game $\mathbf{G}_5$: Finally, one chooses $a^+ \xleftarrow{R} \mathbf{C}$, according the following distribution: $R^+ \xleftarrow{R} \mathbf{M}, r^+ \xleftarrow{R} \mathbf{R}, a^+ \leftarrow \mathcal{E}_{\mathsf{pk}}(R^+, r^+)$. This implicitly defines one pair (R^+, r^+), but the latter is unknown to the simulator.

►**Rule Chal−Hash**$^{(5)}$
> The three values $a^+ \xleftarrow{R} \mathbf{C}$, $g^+ \xleftarrow{R} \{0,1\}^\ell$ and $h^+ \xleftarrow{R} \{0,1\}^\kappa$ have been chosen/given ahead of time, then set $a^\star = a^+$, $\quad b^\star = m^\star \oplus g^+$, $\quad c^\star = h^+$.

The two games $\mathbf{G}_5$ and $\mathbf{G}_4$ are perfectly indistinguishable.

Game $\mathbf{G}_6$: In the simulation of the decryption oracle, we may reject even earlier, if the corresponding R has not been queried to $\mathcal{G}$:

►**Rule Decrypt−R**$^{(6)}$

Look up in G-List for (R, g) such that $R = \mathcal{D}_{\mathsf{sk}}(a)$ (using the PCA-oracle). If no R is found, then output "Reject".

Note that this game differs from the analogous one for the first generic construction BR, because the encryption function is not deterministic, as was the permutation f. Such a simulation differs from the one in the previous game if the value (R, m, a, b) has been queried to $\mathcal{H}$, while $\mathcal{G}(R)$ is unpredictable, and thus $m = \mathcal{G}(R) \oplus b$ in unpredictable too:

$$|\Pr[\mathsf{AskR}_6] - \Pr[\mathsf{AskR}_5]| \leq \frac{q_h}{2^\ell}. \tag{37}$$

One may now note that the event AskR_6 leads to the plaintext R^+ of a^+ by $\mathbf{S}$ in the queries asked to $\mathcal{G}$ and $\mathcal{H}$. By checking all of them, one gets it:

$$\Pr[\mathsf{AskR}_6] \leq \mathsf{Succ}_{\mathbf{S}}^{\mathsf{ow-pca}}(\tau + (q_g + q_h)T_{\mathsf{pca}}). \tag{38}$$

This construction is very generic, and achieves practical security.

Hybrid REACT In this REACT conversion, one can even improve efficiency, replacing the one-time pad [87] by any symmetric encryption scheme: indeed, we have computed some $b = m \oplus K$, where $K = \mathcal{G}(R)$ can be seen as a session key used in a one-time pad encryption scheme. But one could use any symmetric encryption scheme $(\mathbf{E}, \mathbf{D})$ that is just semantically secure

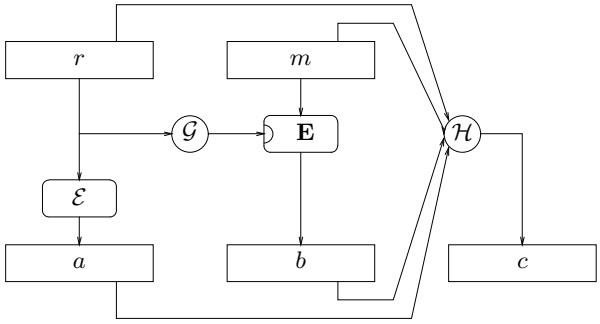

Fig. 15. Hybrid Rapid Enhanced-security Asymmetric Cryptosystem Transform

(under no plaintext nor ciphertext attacks). Indeed, the one-time pad achieves perfect semantic security, against this kind of very weak attacks. But one can tolerate some imperfection. Anyway, most of the candidates to the AES process (the call for symmetric encryption schemes, from the NIST, to become the new international standard), and the AES itself (the winner), resisted to more powerful attacks, and thus can be considered strongly secure in our scenario. Therefore, plaintexts of any size could be encrypted using this conversion (see Figure 15), with a very high speed rate.

7 Conclusion

Recently, Cramer and Shoup proposed the first schemes, for both encryption [23] and signature [24], with formal security proofs in the standard model (without any ideal assumption). The encryption scheme achieves IND-CCA under the sole **DDH** assumption, which says that the

DDH problem is intractable. The signature scheme prevents existential forgeries, even against adaptive chosen-message attacks, under the Strong RSA assumption [2, 29], which claims the intractability of the Flexible RSA problem:

> Given an RSA modulus N and any $y \in \mathbb{Z}_N^\star$, produce x and a prime integer e such that $y = x^e \bmod N$.

Both schemes are very nice because they are the first efficient schemes with formal security proofs in the standard model, but under stronger computational assumptions. We have not presented them, nor the reductions either, which can be found in the original papers. Actually, they are intricate and pretty expensive. Indeed, the complexity of the reductions make them meaningful for large parameters only.

Furthermore, as already noted, no ideal assumptions (such as the random-oracle model) are required, but stronger computational assumptions are needed: the final decision for the best for practical use is not easy.

Moreover, even if the schemes are much more efficient than previous proposals in the standard model, they are still more than twice as expensive as the schemes presented along this paper, in the random-oracle model. This is enough to rule them out from most of the practical applications. Indeed, everybody wants security, but only if it is quite transparent. Therefore, provable security must not decrease efficiency. It is the reason why strong security arguments (which are in an ideal model, but this can be seen as realistic restrictions on the adversary's capabilities) for efficient schemes have a more practical impact than security proofs in the standard model for less efficient schemes.

Of course, quite efficient schemes with formal security proofs are still the target, and thus an exciting challenge.

Acknowledgment

These notes are based on several of my papers, for both signature [67, 68] and encryption [7, 59, 32, 33], written in collaboration with many co-authors, and I would like to take the opportunity of thanking all of them: Mihir Bellare, Anand Desai, Eiichiro Fujisaki, Tatsuaki Okamoto, Philip Rogaway and Jacques Stern. I also warmly thank Benoît Chevallier-Mames, Javier Herranz Sotoca and Duong Hieu Phan for their useful comments on preliminary versions of these notes.

References

1. American National Standards Institute. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm. ANSI X9.62-1998. January 1999.
2. N. Barić and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. In *Eurocrypt '97*, LNCS 1233, pages 480–484. Springer-Verlag, Berlin, 1997.
3. O. Baudron, D. Pointcheval, and J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In *Proc. of the 27th ICALP*, LNCS 1853, pages 499–511. Springer-Verlag, Berlin, 2000.
4. M. Bellare. Practice-Oriented Provable Security. In *ISW '97*, LNCS 1396. Springer-Verlag, Berlin, 1997.
5. M. Bellare, A. Boldyreva, and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In *Eurocrypt '00*, LNCS 1807, pages 259–274. Springer-Verlag, Berlin, 2000.
6. M. Bellare, A. Boldyreva, and A. Palacio. A Separation between the Random-Oracle Model and the Standard Model for a Hybrid Encryption Problem, 2003. Cryptology ePrint Archive 2003/077.
7. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
8. M. Bellare and A. Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *Crypto '02*, LNCS 2442, pages 162–177. Springer-Verlag, Berlin, 2002.

9. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Eurocrypt '00*, LNCS 1807, pages 139–155. Springer-Verlag, Berlin, 2000.
10. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
11. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
12. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996.
13. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Crypto '97*, LNCS 1294, pages 513–525. Springer-Verlag, Berlin, 1997.
14. D. Bleichenbacher. Generating El Gamal Signatures without Knowing the Secret Key. In *Eurocrypt '96*, LNCS 1070, pages 10–18. Springer-Verlag, Berlin, 1996.
15. D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In *Crypto '98*, LNCS 1462, pages 1–12. Springer-Verlag, Berlin, 1998.
16. D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Eurocrypt '97*, LNCS 1233, pages 37–51. Springer-Verlag, Berlin, 1997.
17. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *PKC '00*, LNCS 1751, pages 276–292. Springer-Verlag, Berlin, 2000.
18. D. R. L. Brown and D. B. Johnson. Formal Security Proofs for a Signature Scheme with Partial Message Recovery. In *CT – RSA '01*, LNCS 2020, pages 126–142. Springer-Verlag, Berlin, 2001.
19. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, New York, 1998.
20. S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, Ch. Putnam, Cr. Putnam, and P. Zimmermann. Factorization of a 512-bit RSA Modulus. In *Eurocrypt '00*, LNCS 1807, pages 1–18. Springer-Verlag, Berlin, 2000.
21. B. Chor and R. L. Rivest. A Knapsack Type Public Key Cryptosystem based on Arithmetic in Finite Fields. In *Crypto '84*, LNCS 196, pages 54–65. Springer-Verlag, Berlin, 1985.
22. J.-S. Coron. On the Exact Security of Full-Domain-Hash. In *Crypto '00*, LNCS 1880, pages 229–235. Springer-Verlag, Berlin, 2000.
23. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Crypto '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.
24. R. Cramer and V. Shoup. Signature Scheme based on the Strong RSA Assumption. In *Proc. of the 6th CCS*, pages 46–51. ACM Press, New York, 1999.
25. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
26. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
27. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
28. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987.
29. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *Crypto '97*, LNCS 1294, pages 16–30. Springer-Verlag, Berlin, 1997.
30. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999.
31. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.
32. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '01*, LNCS 2139, pages 260–274. Springer-Verlag, Berlin, 2001. Also appeared as *RSA–OAEP is Still Alive* in the Cryptology ePrint Archive 2000/061. November 2000.
Available from `http://eprint.iacr.org/`.
33. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 17(2):81–104, 2004.
34. O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.
35. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
36. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, New York, 1985.
37. S. Goldwasser, S. Micali, and R. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984.
38. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.

39. C. Hall, I. Goldberg, and B. Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. In *Proc. of ICICS '99*, LNCS, pages 2–12. Springer-Verlag, 1999.
40. J. Håstad. Solving Simultaneous Modular Equations of Low Degree. *SIAM Journal of Computing*, 17:336–341, 1988.
41. A. Joux and R. Lercier. Improvements to the general Number Field Sieve for discrete logarithms in prime fields. *Mathematics of Computation*, 2000. to appear.
42. M. Joye, J. J. Quisquater, and M. Yung. On the Power of Misbehaving Adversaries and Security Analysis of the Original EPOC. In *CT – RSA '01*, LNCS 2020, pages 208–222. Springer-Verlag, Berlin, 2001.
43. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. Submission to IEEE P1363a. August 1998.
Available from `http://grouper.ieee.org/groups/1363/`.
44. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Crypto '96*, LNCS 1109, pages 104–113. Springer-Verlag, Berlin, 1996.
45. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Crypto '99*, LNCS 1666, pages 388–397. Springer-Verlag, Berlin, 1999.
46. A. Lenstra and H. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.
47. A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In *PKC '00*, LNCS 1751, pages 446–465. Springer-Verlag, Berlin, 2000.
48. H.W. Lenstra. On the Chor-Rivest Knapsack Cryptosystem. *Journal of Cryptology*, 3:149–155, 1991.
49. J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In *Crypto '01*, LNCS 2139, pages 230–238. Springer-Verlag, Berlin, 2001.
50. G. Miller. Riemann's Hypothesis and Tests for Primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
51. D. M'Raïhi, D. Naccache, D. Pointcheval, and S. Vaudenay. Computational Alternatives to Random Number Generators. In *Fifth Annual Workshop on Selected Areas in Cryptography (SAC '98)*, LNCS 1556, pages 72–80. Springer-Verlag, Berlin, 1998.
52. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, New York, 1990.
53. V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
54. J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *Crypto '02*, LNCS 2442, pages 111–126. Springer-Verlag, Berlin, 2002.
55. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
56. NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
57. NIST. Descriptions of SHA–256, SHA–384, and SHA–512. Available from `http://www.nist.gov/sha/`, October 2000.
58. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto '98*, LNCS 1462, pages 354–369. Springer-Verlag, Berlin, 1998.
59. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '01*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.
60. T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC '01*, LNCS 1992. Springer-Verlag, Berlin, 2001.
61. D. Pointcheval. *Les Preuves de Connaissance et leurs Preuves de Sécurité*. PhD thesis, université de Caen, December 1996.
62. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
63. D. Pointcheval. About Generic Conversions from any Weakly Secure Encryption Scheme into a Chosen-Ciphertext Secure Scheme. In *Proceedings of the Fourth Conference on Algebraic Geometry, Number Theory, Coding Theory and Cryptography*, pages 145–162, Tokyo, Japan, 2001.
64. D. Pointcheval. Practical Security in Public-Key Cryptography. In *Proc. of ICISC '01*, LNCS 2288. Springer-Verlag, Berlin, 2001.
65. D. Pointcheval. How to Encrypt Properly with RSA. *CryptoBytes*, 5(1):10–19, winter/spring 2002.
66. D. Pointcheval. *Le chiffrement asymétrique et la sécurité prouvée*. PhD thesis, université de Paris VII, May 2002. Thèse d'habilitation.
67. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398. Springer-Verlag, Berlin, 1996.
68. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
69. D. Pointcheval and S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical Report LIENS-96-17, LIENS, October 1996.
70. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.

71. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.
72. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992.
73. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
74. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.
Available from `http://www.rsa.com/rsalabs/pubs/PKCS/`.
75. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
76. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
77. C. P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In *Asiacrypt '00*, LNCS 1976, pages 458–469. Springer-Verlag, Berlin, 2000.
78. D. Shanks. Class Number, a Theory of Factorization, and Genera. In *Proceedings of the Symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971.
79. H. Shimizu. On the Improvement of the Håstad Bound. In *1996 IEICE Fall Conference*, Volume A-162, 1996. In Japanese.
80. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266. Springer-Verlag, Berlin, 1997.
81. V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, december 2001. ISO/IEC JTC 1/SC27.
82. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/060. November 2000.
Available from `http://eprint.iacr.org/`.
83. V. Shoup. OAEP Reconsidered. *Journal of Cryptology*, 15(4):223–249, September 2002.
84. J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart. Flaws in Applying Proof Methodologies to Signature Schemes. In *Crypto '02*, LNCS 2442, pages 93–110. Springer-Verlag, Berlin, 2002.
85. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC '98*, LNCS. Springer-Verlag, Berlin, 1998.
86. S. Vaudenay. Cryptanalysis of the Chor-Rivest Scheme. In *Crypto '98*, LNCS 1462, pages 243–256. Springer-Verlag, Berlin, 1998.
87. G. S. Vernam. Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications. *Journal of the American Institute of Electrical Engineers*, 45:109–115, 1926.