Journal of Telecommunications and Information Technology. Volume 4/2002. Pages 41–56.

Asymmetric Cryptography and Practical Security

David Pointcheval

LIENS – CNRS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France.
David.Pointcheval@ens.fr, http://www.di.ens.fr/users/pointche.

Abstract. Since the appearance of public-key cryptography in Diffie-Hellman seminal paper, many schemes have been proposed, but many have been broken. Indeed, for many people, the simple fact that a cryptographic algorithm withstands cryptanalytic attacks for several years is 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, very few practical schemes can be proven in this so-called "standard model" because such a security level rarely meets with efficiency. A convenient but recent way to achieve some kind of validation of efficient schemes has been to identify some concrete cryptographic objects with ideal random ones: hash functions are considered as behaving like random functions, in the so-called "random oracle model", block ciphers are assumed to provide perfectly independent and random permutations for each key in the "ideal cipher model", and groups are used as black-box groups in the "generic model".
In this paper, we focus on practical asymmetric protocols together with their "reductionist" security proofs. 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.

Keywords: Cryptography, Digital Signatures, Public-Key Encryption, Provable Security, Random Oracle Model

1 Introduction

1.1 Motivation

Since the beginning of public-key cryptography, with the seminal Diffie-Hellman paper [24], many suitable algorithmic problems for cryptography have been proposed (*e.g.* one-way —possibly trapdoor— functions) and many cryptographic schemes have been designed, together with more or less heuristic proofs of their security relative to the intractability of these problems (namely from the number theory, such as the integer factorization, RSA [69], the discrete logarithm [26] and the Diffie-Hellman [24] problems, or from the complexity theory with some $\mathcal{NP}$-complete problems, such as the knapsack [20] problem or the decoding problem of random linear codes [49]). However, most of those schemes have thereafter been broken.

The simple fact that a cryptographic algorithm withstands cryptanalytic attacks for several years is often considered as a kind of validation procedure, but some schemes take a long time before being broken. The best example is certainly the Chor-Rivest cryptosystem [20, 47], based on the knapsack problem, which took more than 10 years to be totally broken [80], whereas before this last attack it was believed to be very hard, since all the classical techniques against the knapsack problems, such as LLL [46], had failed because of the high density of the involved instances. With this example, but also many others, the lack of attacks at some time should never be considered as a security validation of the proposal.

1.2 Provable Security and Practical 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. A 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, and thus they were trying to minimize the required assumptions on the primitives (one-way functions or permutations, possibly trapdoor, etc) [35, 33, 52, 67]. Therefore, they just needed 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 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 used, under the assumption that no polynomial time algorithm exists to solve the underlying problem.

For a few years, more efficient reductions were expected, under the denominations of either "exact security" [10] or "concrete security" [57], 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, 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 [27], and formalized by Bellare and Rogaway [8]. Similarly, block ciphers are identified with families of truly random permutations in the "ideal cipher model" [7]. A few years ago, another kind of idealization was introduced in cryptography, the black-box group [53, 77], where the group operation 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 [73, 17] even use many models together to provide some new validations.

1.3 Outline of the Paper

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). 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.

2 Security Proofs and Security Arguments

2.1 Basic Tools

For asymmetric cryptography, and symmetric cryptography as well, no security can be unconditionally guaranteed, except with the one-time pad [81, 75], an unpractical symmetric encryption method. Therefore, for any cryptographic protocol, security 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 preimage 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, we would like that security only relies on them. 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.

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, *a.k.a.* efficient algorithm, to solve the latter problem, one can polynomially solve the former one, and thus efficiently as well.

Therefore, in order to prove the security of a cryptographic protocol, one first need 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 [33] and signature [35, 36], and provably secure schemes have been suggested. However, those proofs had only a theoretical impact, because both the proposed schemes and the reductions were completely unpractical. Indeed, the reductions were 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 this 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 $2^{100}k^{10}$, 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, it provides an algorithm that requires 2^{130} basic operations, which is much more than the complexity of the best current algorithm, such as NFS [44], which needs less than 2^{100}. Therefore, such a reduction would just be meaningful for numbers above 2048 bits. Concrete examples are given later.

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

- of the hash function, in the "random oracle model";
- of the symmetric block cipher, in the "ideal cipher model";
- of the group, in the "generic model".

The "random oracle model" was the first to be introduced in the cryptographic community [27, 8], and has already been widely accepted. Therefore, in the sequel, we focus on security analyses in this model.

2.3 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 H (such as MD5 [68] or the American standards SHA-1 [55], SHA-256, SHA-384 and SHA-512 [56]). 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 H behaves like a random function. First, Fiat and Shamir [27] applied it heuristically to provide a signature scheme "as secure as" factorization. Then, Bellare and Rogaway [8, 9] formalized this concept in many fields of cryptography: 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 a theoretical counter-example [18] on a clearly wrong design for practical purpose! Therefore, this model has been strongly accepted by the community, and is considered as a good one, in which proofs of security 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".

More formally, 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.

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 [32, 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.

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 [36]).

Definitions A signature scheme is defined by the three following algorithms:

- The *key generation algorithm* 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 K produces a pair $(\mathsf{k_p}, \mathsf{k_s})$ of matching public and private keys. Algorithm 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 a security level in 2^k (the expected minimal time complexity of any attack).
- The *signing algorithm* Σ. Given a message m and a pair of matching public and private keys $(\mathsf{k_p}, \mathsf{k_s})$, Σ produces a signature σ. The signing algorithm might be probabilistic.
- The *verification algorithm* V. Given a signature σ, a message m and a public key $\mathsf{k_p}$, V tests whether σ is a valid signature of m with respect to $\mathsf{k_p}$. 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*.

In many cases this 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 her into her forgery. We focus on two specific kinds of attacks against signature schemes: the *no-message attacks* and the *known-message attacks*. In the first scenario, the attacker only knows the public key of the signer. In the second one, 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 the *adaptive chosen-message attack*, where the attacker can ask the signer to sign any message of her choice. She can therefore adapt her queries according to previous answers.

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

$$\mathsf{Succ}_{\mathbf{A}} = \Pr\left[(\mathsf{k_p}, \mathsf{k_s}) \leftarrow K(1^k), (m, \sigma) \leftarrow \mathbf{A}^{\Sigma_{\mathsf{k_s}}}(\mathsf{k_p}) : V(\mathsf{k_p}, m, \sigma) = 1\right].$$

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 $\mathbf{A}^{\Sigma_{\mathsf{k_s}}}$. Of course, in its answer, there is the natural restriction that the returned signature has not been obtained from the signing oracle $\Sigma_{\mathsf{k_s}}$ itself.

This above security level is the strongest one that one can formalize in the communication model we consider. We insist on the fact that in the current communication model, we give the adversary complete access to the cryptographic primitive, but as a black-box. She can ask any query of her choice, and the box always answers correctly, in constant time. Such a model does not consider timing attacks [42], 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 [43], or to make the primitive fail on some computation [11, 15]. They are not captured either by this model.

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 is defined by the three following algorithms:

- The *key generation algorithm* K. On input 1^k where k is the security parameter, the algorithm K produces a pair $(\mathsf{k_p}, \mathsf{k_s})$ of matching public and private keys. Algorithm K is probabilistic.
- The *encryption algorithm* E. Given a message m and a public key $\mathsf{k_p}$, E produces a ciphertext c of m. This algorithm may be probabilistic. We write $E(\mathsf{k_p}, m; r)$ where r, in the probabilistic case, is the random input to E.
- The *decryption algorithm* D. Given a ciphertext c and the private key $\mathsf{k_s}$, D 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 $\mathbf{A}$, her success in inverting 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:

$$\mathsf{Succ}_{\mathbf{A}} = \Pr_{m,r}[(\mathsf{k_p}, \mathsf{k_s}) \leftarrow K(1^k) : \mathbf{A}(\mathsf{k_p}, E(\mathsf{k_p}, m; r)) = m].$$

However, many applications require more from an encryption scheme, namely the *semantic security* (IND) [33], *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, one of which has been encrypted, with a probability significantly better than one half: her advantage $\mathsf{Adv}_{\mathbf{A}}$, formally defined as

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

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

A later notion is *non-malleability* (NM) [25]. 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 [6] (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 her choice, granted the public key, hence the *chosen-plaintext attack* (CPA). She may furthermore have access to more information, modeled by partial or full access to some oracles: a plaintext-checking oracle which, on input a pair (m, c), answers whether c encrypts the message m. This attack has been named the *Plaintext-Checking Attack* (PCA) [58]; a validity-checking oracle which, on input a ciphertext c, just answers whether it is a valid ciphertext or not. This weak oracle had been enough to break some famous encryption schemes [13, 40], running the so-called reaction attacks [37]; or the decryption oracle itself, which on any ciphertext, except the challenge ciphertext, answers the corresponding plaintext (*non-adaptive [52]/adaptive [67] chosen-ciphertext attacks*). This latter scenario which allows adaptively chosen ciphertexts as queries to the decryption oracle is the strongest attack, and is 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 [38, 76], namely with a small encryption exponent, using the Chinese Remainders Theorem. But recent results prove that semantically secure schemes, in the classical sense as described above, remain secure in multi-user scenarios [5, 3], whatever the kind of attacks.

A general study of these security notions and attacks was conducted in [6], we therefore refer the reader to this paper for more details. See also the summary diagram

$$\begin{tikzcd}
\mathsf{NM\text{-}CPA} \arrow[d] & & \mathsf{NM\text{-}CCA} \arrow[ll] \arrow[d, shift right] \\
\mathsf{IND\text{-}CPA} \arrow[d] & & \mathsf{IND\text{-}CCA} \arrow[ll] \arrow[u, shift right] \arrow[d] \\
\mathsf{OW\text{-}CPA} & \mathsf{OW\text{-}PCA} \arrow[l] & \mathsf{OW\text{-}CCA} \arrow[l]
\end{tikzcd}$$

OW	– One-Wayness
IND	– Indistinguishability of Encryptions (*a.k.a.*, Semantic Security)
NM	– Non-Malleability
CPA	– Chosen-Plaintext Attack
PCA	– Plaintext-Checking Attack
CCA	– Chosen-Ciphertext Attack

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

on Figure 1. However, we can just review the scenarios we will be interested in 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 her choice, has been encrypted, while she can ask any query she wants to a decryption oracle (except the challenge ciphertext). This is the strongest scenario one can define for encryption (still in our communication model), and thus our goal when we design a cryptosystem.

4 The Basic 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 [69];
2. the schemes based on the discrete logarithm problem, and on the Diffie-Hellman problems [24], in any "suitable" group. The first groups in use were cyclic subgroups of $\mathbf{Z}_p^\star$, the multiplicative group of the modular quotient ring $\mathbf{Z}_p = \mathbf{Z}/p\mathbf{Z}$. 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 [54] 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 [44] has a 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 last record, in august 1999, by factoring a 155-digit integer, product of two 78-digit primes [19].

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 =
  109417386415705274218097073220403576120037329454492059909138421314763499842889\
  347847179972578912673324976257528997818337970765372440271467435315933543338 97
= 102639592829741105772054196573991675900716567808038066803341933521790711307779
* 106603488380168454820927220360012878679207958575989291522270608237193062808643
```

However, this record required thousands of machines, and three months of computation. Furthermore, due to the above complexity of NFS, integer factoring is believed to be a difficult problem, especially for products of two primes of similar sizes larger than 384 bits.

Unfortunately, it 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 $\mathbf{Z}_n$ which is isomorphic to the product ring $\mathbf{Z}_p \times \mathbf{Z}_q$ 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 powers of 2 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 $\mathbf{Z}_n^\star$ of $\mathbf{Z}_n$. 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 [69] 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 $y \in \mathbf{Z}_n^\star$, compute the modular e-th root x of y (*i.e.* $x \in \mathbf{Z}_n^\star$ 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. However nobody knows whether the factorization is required, but nobody knows how to do without it either:

> **The RSA Assumption.** For any product of two large 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 $(\mathbf{Z}_q, +)$, a subgroup of $(\mathbf{Z}_p^\star, \times)$ for $q|p-1$, or 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 $x \in \mathbf{Z}_q$. But such a $\mathbf{g}$ could be an element in $\mathbf{Z}_p^\star$ 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 $x \in \mathbf{Z}_q$ 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 $z \in \mathbf{Z}_q$, 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.

Very recently, Tatsuaki Okamoto and the author [60] defined a new variant of the Diffie-Hellman problem, which we called the *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 this 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 $\mathbf{g}$ [74, 65]. 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 $\mathbf{Z}_p^\star$, 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 [39] has a 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 modulo a 120-digit prime. 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.

The **CDH** problem is usually believed to be much stronger than the **DDH** problem, which means that the **GDH** problem is difficult. This was the motivation of our work on new encryption schemes based on the **GDH** problem [58] (see section 5.3).

5 Provably Secure Designs

5.1 Introduction

Until 1996, no practical **DL**-based cryptographic scheme has ever been formally studied, only heuristically. And surprisingly, at the Eurocrypt '96 conference, two opposite studies were conducted on the El Gamal signature scheme [26], the first **DL**-based signature scheme designed in 1985 and depicted on Figure 2.

Initialization
g a generator of $\mathbf{Z}_p^\star$, where p is a large prime
K**: Key Generation**
private key $x \in \mathbf{Z}_{p-1}^\star$ public key $y = g^x \bmod p$ $\rightarrow (y, x)$
Σ**: Signature of** $m \rightarrow (r, s)$
K is randomly chosen in $\mathbf{Z}_{p-1}^\star$ $r = g^K \bmod p \qquad s = (m - xr)/K \bmod p - 1$ $\rightarrow (r, s)$ is a signature of m
V**: Verification of** (m, r, s)
check whether $g^m \overset{?}{=} y^r r^s \bmod p$

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 [12], showed such a universal forgery when the generator g is not properly chosen. The second one, from Jacques Stern and the author [62], proved the security, against existential forgeries under adaptive chosen-message attacks, of a slight variant with a randomly chosen generator g. The slight variant simply replaces the message m by $H(m, r)$ in the computation, while one uses a hash function 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. Furthermore, the conclusions have to be strictly followed by developers, otherwise the concrete implementation of a secure scheme can be very weak.

5.2 Digital Signature Schemes

History The first *secure* signature scheme was proposed by Goldwasser *et al.* [35] 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 [24], Rivest, Shamir and Adleman [69] 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 $\mathbf{Z}_N^\star$, 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 [34], and uses hash functions in order to create a kind of virtual verifier. The first application was derived from the Fiat–Shamir [27] zero-knowledge identification protocol, based on the hardness of extracting square roots, with a brief outline of its security. Another famous identification scheme [71], together with the signature scheme [72], 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 $\mathbf{Z}_p^\star$ of order q. It also creates a pair of keys, the private key $x \in \mathbf{Z}_q^\star$ 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 $K \in \mathbf{Z}_q$, the "challenge" $e = H(m, r)$ and $s = K + ex \bmod q$. This latter satisfies $r = g^s y^e \bmod p$ with $e = H(m, r)$, 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.

Secure Designs

Schnorr's Signature and Variants. In our papers [62, 63], with Jacques Stern, we formally proved the above paradigm when 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 $\mathbf{Q}_i$'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 message m is a triple (σ_1, h, σ_2), with $h = H(m, \sigma_1)$ 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

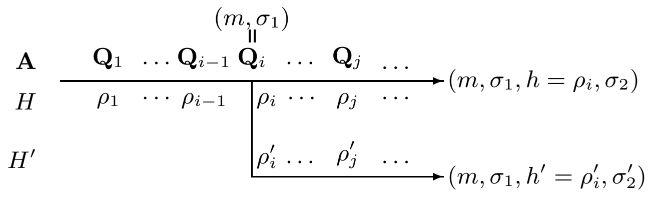
(m, σ_1)
$\mathbf{A}$
$\mathbf{Q}_1 \ldots \mathbf{Q}_{i-1}\ \mathbf{Q}_i \ldots \mathbf{Q}_j \ldots$
$(m, \sigma_1, h = \rho_i, \sigma_2)$
H
$\rho_1 \cdots \rho_{i-1}$
$\rho_i \cdots \rho_j \cdots$
H'
$\rho'_i \ldots \rho'_j \ldots$
$(m, \sigma_1, h' = \rho'_i, \sigma'_2)$

Fig. 3. The Oracle Replay Technique

enough, then with good probability, the adversary is able to answer to many distinct outputs from the 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 signa-

Initialization (security parameter k)
$\mathbf{g}$ a generator of any cyclic group $(\mathcal{G}, +)$ of order q, with $2^{k-1} \le q < 2^k$ H a hash function: $\{0,1\}^\star \to \mathbf{Z}_q$
K**: Key Generation**
private key $x \in \mathbf{Z}_q^\star$ public key $\mathbf{y} = -x \cdot \mathbf{g}$ $\to (\mathbf{y}, x)$
Σ**: Signature of** $m \to (\mathbf{r}, h, s)$
K is randomly chosen in $\mathbf{Z}_q^\star$ $\mathbf{r} = K \cdot \mathbf{g}$ $\quad h = H(m, r)$ $\quad s = K + xh \bmod q$ $\to (\mathbf{r}, h, s)$ is a signature of m
V**: Verification of** (m, r, s)
check whether $h \stackrel{?}{=} H(m, \mathbf{r})$ and $\mathbf{r} \stackrel{?}{=} s \cdot \mathbf{g} + h \cdot \mathbf{y}$

Fig. 4. The Schnorr Signature Scheme.

tures $(\mathbf{r} = \sigma_1, h, s = \sigma_2)$ and $(\mathbf{r}' = \sigma'_1, h', s' = \sigma'_2)$ of an identical message m such that $\sigma_1 = \sigma'_1$, but $h \neq h'$. We then can extract the discrete logarithm of the public key:

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

which leads to

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

Let use denote by ε the success probability of the adversary in performing an existential forgery after q_h queries to the random oracle H. One can prove that for ε large enough, more precisely $\varepsilon \ge 7q_h/2^k$, after less than $16q_h/\varepsilon$ repetitions of this adversary, one can obtain such a pair of signatures with probability greater than $1/9$.

However, this just covers the no-message attacks, which are the weakest attacks! But because we can simulate any zero-knowledge protocol, even without having to restart the simulation since we are in front of an honest verifier (*i.e.* the challenge is randomly chosen by the random oracle H) one can easily simulate the signer without the private key:

- one first chooses random $h, s \in \mathbf{Z}_q$;
- one computes $\mathbf{r} = s \cdot \mathbf{g} + h \cdot \mathbf{y}$ and defines $H(m, \mathbf{r})$ 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 H, by defining $H(m, \mathbf{r})$ to be equal to h. This simulation is almost perfect since H is supposed to output a random value to any new query, and h is indeed a random value. Nevertheless, if the query $H(m, \mathbf{r})$ has already been asked, $H(m, \mathbf{r})$ is already defined, and thus the definition $H(m, \mathbf{r}) \leftarrow h$ 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: let $\mathbf{A}$ be an adversary against the Schnorr signature scheme, with security parameter k, in the random oracle model. We assume that, after q_h queries to the random oracle and q_s queries to the signing oracle, $\mathbf{A}$ produces, with probability $\varepsilon \geq 10(q_s + 1)(q_s + q_h)/2^k$, a valid signature. Then, after less than $23q_h/\varepsilon$ repetitions of this adversary, one can extract the private key x with probability $\varepsilon' \geq 1/9$.

From a more practical point of view, this result states 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 $207q_hT$.

Brickell, Vaudenay, Yung and the author extended this technique [64, 16] to many variants of El Gamal [26] and DSA [54], such as the Korean Standard KCDSA [41]. 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.

RSA-Based Signatures. Unfortunately, with these signatures, we don't really achieve our goal, because the reduction is costly: if one can break the signature scheme within an expected time T, and q_h queries to the hash function, then one can compute $\log_g y$ within an expected time $207q_hT$, where q_h can be huge, as much as 2^{60} in practice. This security proof is meaningful only for very large groups.

In 1996, Bellare and Rogaway [10] 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 $H : \{0,1\}^\star \to \mathbf{Z}_N^\star$, and computes the e^{th} root, $\sigma = H(m)^d \bmod N$. The verification algorithm simply checks whether the following equality holds, $H(m) = \sigma^e \bmod N$. For this scheme, they proved, in the random oracle model: if an adversary 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 RSA function can be inverted with probability ε' within time t' where

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

with T_{exp} the time for an exponentiation to the power e, modulo N. This reduction has been recently improved [21], thanks to the random self-reducibility of the RSA function, into:

$$\varepsilon' \geq \frac{\varepsilon}{q_s} \times \exp(-1) \quad \text{ and } \quad t' \leq t + (q_s + q_h)T_{exp}.$$

This latter reduction works as follows: we assume the existence of an adversary that produces an existential forgery, with probability ε, within time t, after q_h queries to the random oracle H and q_s queries to the signing oracle. We provide this adversary with all the inputs/outputs he needs, while trying to extract the e-th root of a given y. For that, we have to simulate the random oracle and the signing oracle.

Simulation of the random oracle H: For any fresh query m to H, one chooses a random $r \in \mathbf{Z}_N^\star$ and flips a biased coin (which returns 0 with probability p, and 1 with probability $1-p$.) If 0 appears, one defines and returns $H(m) = r^e \bmod N$, otherwise one defines and returns $H(m) = yr^e \bmod N$.

Simulation of the signing oracle: For any fresh query m, one first invokes the random oracle on m (if not yet done). If $H(m) = r^e \bmod N$ for some known r, then one returns r as the signature of m, otherwise we stop the simulation and return a failure signal.

At the end of the game, the adversary outputs a valid message/signature $H(m) = \sigma^e \bmod N$. If $H(m)$ has been asked to H during the simulation then, with probability $1-p$, $H(m) = yr^e = \sigma^e \bmod N$ and thus $y = (\sigma/r)^e \bmod N$, which leads to an e-th root of y. Otherwise we return a failure signal.

One must first remark that the H simulation is perfect since a new random element in $\mathbf{Z}_N^\star$ is returned for any new query. However, the signing oracle simulation may fail when a signing query is done on a message such that $H(m) = yr^e \bmod N$. Indeed, in this case, the simulation aborts. But such a case happens with probability $1-p$ for any signature. Therefore, the simulation is perfect with probability p^{q_s}, and in such a good case, the forgery leads to the e-th root of y with probability $1-p$. Therefore, the success probability of our RSA inversion is $(1-p)p^{q_s}\varepsilon$, which is optimal for $p = 1 - 1/(q_s+1)$. And for this parameter, and a huge value q_s, the success probability is approximately ε/eq_s.

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 $q_h \leq 2^{60}$, but q_s can be limited below 2^{30}.

However, one would like to get more, suppressing any coefficient. In their paper [10], Bellare and Rogaway proposed such a better candidate, the Probabilistic Signature Scheme (PSS, see Figure 5): the key generation is still the same, but the signature process involves three hash functions

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

where $k = k_0 + k_1 + k_2 + 1$ is the bit-length of the modulus N. For each message m to be signed, one chooses a random string $r \in \{0,1\}^{k_1}$. One first computes $w = H(m,r)$, $s = G(w) \oplus r$ and $t = F(w)$. 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 e^{th} root, $\sigma = y^d \bmod N$.

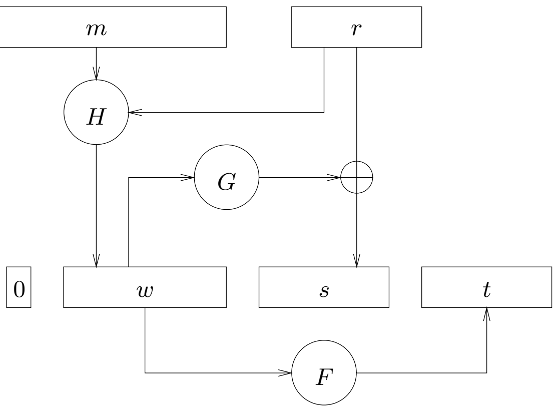

Fig. 5. Probabilistic Signature Scheme

The verification algorithm first computes $y = \sigma^e \bmod N$, and parses it as $y = b\|w\|s\|t$. Then, one can get $r = s \oplus G(w)$, and checks whether $b = 0$, $w = H(m, r)$ and $t = F(w)$.

About PSS, Bellare and Rogaway proved, in the random oracle model: if an adversary 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 functions (F, G and H altogether) and the signing oracle respectively, then the RSA function can be inverted with probability ε' within time t' where

$$\begin{aligned} & \varepsilon' \geq \varepsilon - \frac{1}{2^{k_2}} - (q_s + q_H) \cdot \left(\frac{q_s}{2^{k_1}} + \frac{q_h + q_s + 1}{2^{k_2}} \right) \\ \text{and} \quad & t' \leq t + (q_s + q_H) k_2 T_{exp}, \end{aligned}$$

with T_{exp} the time for an exponentiation to the power e, modulo N.

The reduction is a bit more intricate than the previous one: once again, we assume the existence of an adversary that produces an existential forgery, with probability ε, within time t, after q_F, q_G, q_H queries to the random oracles F, G, H (we denote $q_h = q_F + q_G + q_H$) and q_s queries to the signing oracle. We provide this adversary with all the inputs/outputs he needs, while trying to extract the e-th root of a given y. For that, we have to simulate the random oracles and the signing oracle. For any fresh query (m, r) to the random oracle H, one chooses a random $u \in \mathbf{Z}_N^\star$ and computes $z = yu^e \bmod N$, until the most significant bit of z is 0. Then one parses z into $0\|w\|s\|t$. Then one defines $H(m, r) \leftarrow w$, $G(w) \leftarrow s \oplus r$ and $F(w) \leftarrow t$. One finally returns w.

For any query w to the random oracles F or G, if the answer has not already been defined (during a H simulation) then a random string is returned.

A signing query m is trivially answered: one chooses a random r and runs a specific simulation of $H(m, r)$: one chooses a random $u \in \mathbf{Z}_N^\star$ and computes $z = u^e \bmod N$, until the most significant bit of z is 0. Then one parses z into $0\|w\|s\|t$, and defines $H(m, r) \leftarrow w$, $G(w) \leftarrow s \oplus r$ and $F(w) \leftarrow t$. One finally returns u as a signature of m.

At the end of the game, the adversary outputs a valid message/signature (m, σ), where $\sigma^e = 0\|w\|s\|t \bmod N$, which corresponds to the signature of m with the random $r = G(w) \oplus s$. If $H(m, r)$ has not been asked, $H(m, r) = w$ with probability $1/2^{k_2}$. Therefore

$$\Pr[\mathsf{valid}(m, \sigma) \mid \neg \mathsf{AskH}(m, r)] \leq 2^{-k_2},$$

where "$\mathsf{valid}(m,\sigma)$" denotes the event that the message/signature (m,σ) is a valid one (and thus accepted by the verification algorithm), and "$\mathsf{AskH}(m,r)$" denotes the event that the query (m,r) has been asked to the random oracle H.

Since this is a forgery, m has never been signed by the signing oracle, and thus $H(m,r)$ has been asked directly by the adversary: $H(m,r) \leftarrow w$, $G(w) \leftarrow s \oplus r$ and $F(w) \leftarrow t$, where $yu^e = 0\|w\|s\|t$, which leads to an e-th root of y.

However, the simulations may not be perfect:

- the random oracles simulations may fail if when defining $F(w)$ and $G(w)$, during an H-simulation (a direct one, or the simulation done for the signing simulation), one of them had already been defined before. But this may just occur with probability less than $q_F \cdot 2^{-k_2}$ (because of a previous direct query to F), $q_G \cdot 2^{-k_2}$ (because of a previous direct query to G) or $(q_H + q_s) \cdot 2^{-k_2}$ (because of a previous direct or indirect H-simulation).
- even after many iterations, the z (computed during the H-simulation, or the signing simulation) may still be greater than $N/2$. We limit this number of iterations to k_2. Then the probability for z to be still greater than $N/2$ is less than $1/2^{k_2}$.
- the signing simulation may fail if the $H(m,r)$ value has already been defined. But this may only occur with probability $(q_H + q_s)2^{-k_1}$.

Therefore, the global success probability in inverting y is greater than

$$\varepsilon - 2^{-k_2} - (q_H + q_s)\left[\frac{q_F + q_G}{2^{k_2}} + \frac{q_H + q_s}{2^{k_2}} + \frac{1}{2^{k_2}} + \frac{q_s}{2^{k_1}}\right],$$

hence the result.

As fas as time complexity is concerned, each H simulation (which can be launched by a signing simulation) requires at most k_2 modular exponentiations to the power e, hence the result. Thanks to this exact and efficient security result, RSA–PSS has become the new PKCS #1 v2.0 standard for signature [70]. Another variant has been proposed with message-recovery. PSS-R 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 this latter is inside the former one.

5.3 Public-Key Encryption

History

The RSA Encryption Scheme. In the same paper as the RSA signature scheme [69], 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 $\mathbf{Z}_N^\star$, 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 $\mathsf{OW\text{-}CPA}$, relative to the RSA problem. The determinism makes a plaintext-checking oracle useless. Indeed, the encryption of a message m, under a public key $\mathsf{k_p}$ 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 $\mathsf{OW\text{-}PCA}$ 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 $\mathbf{Z}$ and therefore leads to an easy recovery of m.

The El Gamal Encryption Scheme. In 1985, El Gamal [26] also designed a public-key encryption scheme based on the Diffie-Hellman key exchange protocol [24]: given a cyclic group $\mathcal{G}$ of order prime q and a generator $\mathbf{g}$, the generation algorithm produces a random element $x \in \mathbf{Z}_q^\star$ 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 $(\mathbf{c} = a \cdot \mathbf{g}, \mathbf{d} = a \cdot \mathbf{y} + \mathbf{m})$. 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 [79]. For OW-PCA, it relies on the new Gap Diffie-Hellman problem [60].

Secure Designs 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?

OAEP: the Optimal Asymmetric Encryption Padding. In 1994, Bellare and Rogaway proposed a generic conversion [9], in the random oracle model, the "Optimal Asymmetric Encryption Padding" (OAEP, see Figure 6), which was claimed to apply to

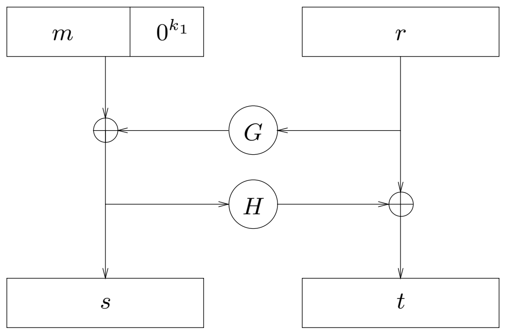

Fig. 6. Optimal Asymmetric Encryption Padding

any family of trapdoor one-way permutations, such as RSA. The key generation produces a one-way permutation $f : \{0,1\}^k \to \{0,1\}^k$, the public key. The private key is the inverse permutation g, which requires a trapdoor to be computable. The scheme involves two hash functions

$$G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}, \quad H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0},$$

where $k = k_0 + k_1 + n + 1$. For any message $m \in \{0,1\}^n$ to be encrypted. Instead of computing $f(m)$, as done with the above plain-RSA encryption, one first modifies m, choosing a random string $r \in \{0,1\}^{k_0}$. Then one computes $s = (m\|0^{k_1}) \oplus G(r)$ and $t = r \oplus H(s)$. Finally, one computes $c = f(s\|t)$.

The decryption algorithm first computes $P = g(c)$, granted the private key g, and parses it as $P = s\|t$. Then, one can get $r = t \oplus H(s)$, and $M = s \oplus G(r)$, which is finally parsed into $M = m\|0^{k_1}$, if the k_1 least significant bits are all 0.

For a long time, the OAEP conversion has been widely believed to provide an IND-CCA encryption scheme from any trapdoor one-way permutation. However, the sole proven result (weak plaintext-awareness [9]) was the semantic security against non-adaptive chosen-ciphertext attacks (*a.k.a.* lunchtime attacks [52]). And recently, Shoup [78] showed that it was very unlikely that a stronger security result could be proven. However, because of the wide belief of a strong security level, RSA–OAEP became the new PKCS #1 v2.0 for encryption [70], and thus a *de facto* standard, after an effective attack against the PKCS #1 v1.5 [13].

Fortunately, Fujisaki, Okamoto, Stern and the author [31] provided a complete security proof: first we proved that combined with a trapdoor partial-domain one-way permutation, the OAEP construction leads to an IND-CCA cryptosystem. A partial-domain one-way permutation f is a one-way permutation such that given $y = f(s\|t)$ it is intractable to recover the full $s\|t$, but even s only. Furthermore, we provided a complete reduction between the full-domain inversion of RSA and the partial-domain inversion. Therefore, RSA–OAEP really achieves IND-CCA security under the RSA assumption.

The proof is a bit intricate, so we refer the reader to [31] for more information. Anyway, we can claim

> if there exists a CCA–adversary against the "semantic security" of RSA–OAEP (with a k-bit long modulus, with $k > 2k_0$), 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 could be solved with probability ε', within time bound t', where
>
> $$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon^2}{4} - \varepsilon \cdot \left(\frac{2q_Dq_G + q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}} + \frac{32}{2^{k-2k_0}} \right) \\ t' &\leq 2t + q_H \cdot (q_H + 2q_G) \times \mathcal{O}(k^3). \end{aligned}$$

Unfortunately, the reduction is very expensive, and is thus meaningful only for huge moduli, 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 [45] 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 [48]).

More General Conversions. 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 [29] 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 5.3), one obtains an IND-CCA encryption scheme relative to the **DDH** problem. Later, independently, Fujisaki and Okamoto [30] and the author [61] 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.

REACT: a Rapid Enhanced-security Asymmetric Cryptosystem Transform. 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 [58]: REACT, for "Rapid Enhanced-security Asymmetric Cryptosystem Transform".

```
$K'$: Key Generation
$(\mathsf{k_p}, \mathsf{k_s}) \leftarrow K(1^k)$
$\rightarrow (\mathsf{k_p}, \mathsf{k_s})$

$E'$: 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
$a = E(\mathsf{k_p}, R; r)$    $b = m \oplus G(R)$    $c = H(R, m, a, b)$
$\rightarrow (a, b, c)$ is the ciphertext

$D'$: Decryption of $(a, b, c)$
Given $a \in \mathbf{C}$, $b \in \{0,1\}^\ell$ and $c \in \{0,1\}^\kappa$
$R = D(\mathsf{k_s}, a)$    $m = b \oplus G(R)$
if $c = H(R, m, a, b)$ and $R \in \mathbf{M} \rightarrow m$ is the plaintext
                    (otherwise, "Reject: invalid ciphertext")
```

Fig. 7. Rapid Enhanced-security Asymmetric Cryptosystem Transform $\mathbf{S}'$

This 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 [58] on any encryption scheme $\mathbf{S} = (K, E, D)$

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

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 G and H,

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

where κ is the security parameter, while ℓ denotes the size of the messages to encrypt. The REACT conversion is depicted on Figure 7.

In this new scheme $\mathbf{S}'$, one can claim that if an attacker, against the semantic security in a chosen-ciphertext scenario, can gain an advantage ε after q_D, q_G and q_H queries to the decryption oracle and to the random oracles G and H respectively, within a time t, then one can design an algorithm that outputs, for any given C, the plaintext of C, after less than $q_G + q_H$ queries to the Plaintext-Checking Oracle with probability greater than $\varepsilon/2 - q_D/2^\kappa$, within a time $t + (q_G + q_H)T_{\mathsf{PCA}}$, where T_{PCA} denotes the times required by the PCA oracle to answer any query.

This security result, in the random oracle model, comes from two distinct remarks:

– the adversary has necessarily asked either $G(R)$ or $H(R, m_i, a, b)$ to get any information about the encrypted message m (either m_0 or m_1). Which means that for a given $C = E(\mathsf{k_p}, R; r)$, R is in the list of queries asked to G or to H. Simply asking for the $q_G + q_H$ candidates to the Plaintext-Checking Oracle, one can output the right one. Then, with probability $\varepsilon/2$, one inverts E, after $(q_G + q_H)$ queries to the Plaintext-Checking Oracle.
– However, in the chosen-ciphertext scenario, the adversary may ask queries to the decryption oracle. We have to simulate this. For each query (a, b, c) asked by the adversary to the decryption oracle, one looks at all the pairs (R, m) such that (R, m, a, b) has been asked to the random oracle H. For any such R, one asks the Plaintext-Checking Oracle whether a is a ciphertext of R (remark that it does not make more queries to the Plaintext-Checking Oracle, since it has already been taken into account above). Then it computes $K = G(R)$, maybe using a simulation of G if the query R has never been asked. If $b = K \oplus m$ then one outputs m as the plaintext of the triple (a, b, c). Therefore, any correctly computed ciphertext is decrypted by the simulator. But if the adversary has not asked $H(R, m, a, b)$ the probability that the ciphertext is valid, and thus the decryption not correctly simulated, is less than $1/2^\kappa$.

Hybrid Conversion. In this REACT conversion, one can improve efficiency, replacing the one-time pad [81] by any symmetric encryption scheme: indeed, we have computed some $b = m \oplus K$, where $K = G(R)$ 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 (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 8), with a very high speed rate.

RSA–OAEP Alternatives. As we have said, RSA–OAEP has become a *de facto* standard, even if its security has recently been subject to controversy. However, the practical security, for usual sizes (between 512 and 1024 bits), is not really proven because of the huge overhead in the reduction. Some alternatives have been proposed, such as OAEP+ [78] and SAEP(+) [14], but still with expensive reductions, in the general RSA context (some efficient reductions have been proposed for OAEP or SAEP, but only with RSA exponent 3, or the Rabin primitive [66]). Therefore RSA–REACT [59]

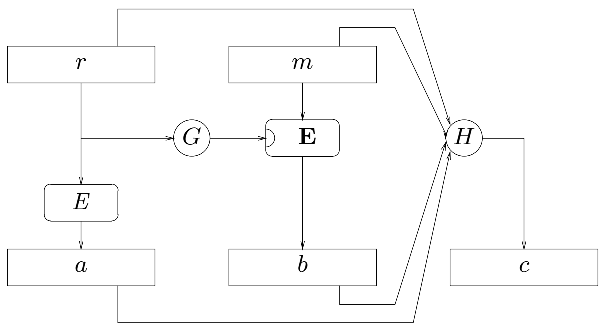

Fig. 8. Hybrid Rapid Enhanced-security Asymmetric Cryptosystem Transform

looks like the best alternative to RSA–OAEP, thanks to the efficient reduction, and the provable security relative the RSA assumption (in the random oracle model).

6 Conclusion

Recently, Cramer and Shoup proposed the first schemes, for both encryption [22] and signature [23], 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, 28], which claims the intractability of the Flexible RSA problem:

Given an RSA modulus N and any $y \in \mathbf{Z}_N^\star$, 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. However, we have not presented them, nor the reductions either. Actually, they are very intricate and pretty expensive. Furthermore, even if no ideal assumptions are required, the complexity of the reductions make them meaningless for practical parameters. Moreover, even if the schemes are much more efficient than previous proposals in the standard model, they are still much 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 practical use. Indeed, everybody wants security, but only if it is quite transparent (and particularily from the financial point of view). Therefore, provable security must not decrease efficiency. It is the reason why strong security arguments, under a realistic restriction 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.

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 W. Fumy, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–484, Konstanz, Germany, 1997. Springer-Verlag, Berlin.

3. O. Baudron, D. Pointcheval, and J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In J. D. P. Rolim U. Montanari and E. Welzl, editors, *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP '2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 499–511, Geneva, Switzerland, 2000. Springer-Verlag, Berlin.
4. M. Bellare. Practice-Oriented Provable Security. In E. Okamoto, G. Davida, and M. Mambo, editors, *Proceedings of First International Workshop on Information Security (ISW '97)*, volume 1396 of *Lecture Notes in Computer Science*. 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 B. Preneel, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274, Brugge, Belgium, 2000. Springer-Verlag, Berlin.
6. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In H. Krawczyk, editor, *Advances in Cryptology – proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Santa-Barbara, California, 1998. Springer-Verlag, Berlin.
7. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Brugge, Belgium, 2000. Springer-Verlag, Berlin.
8. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, U.S.A., 1993. ACM Press, New York.
9. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, 1995. Springer-Verlag, Berlin.
10. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, 1996. Springer-Verlag, Berlin.
11. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B. Kaliski, editor, *Advances in Cryptology – proceedings of CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525, Santa-Barbara, California, 1997. Springer-Verlag, Berlin.
12. D. Bleichenbacher. Generating El Gamal Signatures without Knowing the Secret Key. In U. Maurer, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 10–18, Saragossa, Spain, 1996. Springer-Verlag, Berlin.
13. D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In H. Krawczyk, editor, *Advances in Cryptology – proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Santa-Barbara, California, 1998. Springer-Verlag, Berlin.
14. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In J. Kilian, editor, *Advances in Cryptology – proceedings of CRYPTO '2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291, Santa-Barbara, California, 2001. Springer-Verlag, Berlin.
15. D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In W. Fumy, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51, Konstanz, Germany, 1997. Springer-Verlag, Berlin.
16. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In H. Imai and Y. Zheng, editors, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292, Melbourne, Australia, January 2000. Springer-Verlag, Berlin.
17. D. R. L. Brown and D. B. Johnson. Formal Security Proofs for a Signature Scheme with Partial Message Recovery. In D. Naccache, editor, *The Cryptographers' Track at RSA Conference '2001 (RSA '2001)*, volume 2020 of *Lectures Notes in Computer Science*, pages 126–142, San Francisco , California, USA, April 2001. Springer-Verlag, Berlin.
18. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC '98)*, pages 209–218. ACM Press, New York, 1998.
19. 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 B. Preneel, editor,

Advances in Cryptology – Proceedings of EUROCRYPT '2000, volume 1807 of *Lecture Notes in Computer Science*, pages 1–18, Brugge, Belgium, 2000. Springer-Verlag, Berlin.

20. B. Chor and R. L. Rivest. A Knapsack Type Public Key Cryptosystem based on Arithmetic in Finite Fields. In B. Blakley and D. Chaum, editors, *Advances in Cryptology – Proceedings of CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 54–65, Santa-Barbara, California, 1985. Springer-Verlag, Berlin.
21. J.-S. Coron. On the Exact Security of Full-Domain-Hash. In M. Bellare, editor, *Advances in Cryptology – proceedings of CRYPTO '2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Santa-Barbara, California, 2000. Springer-Verlag, Berlin.
22. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In H. Krawczyk, editor, *Advances in Cryptology – proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa-Barbara, California, 1998. Springer-Verlag, Berlin.
23. R. Cramer and V. Shoup. Signature Scheme based on the Strong RSA Assumption. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 46–51, Singapore, 1999. ACM Press, New York.
24. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
25. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
26. 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.
27. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In A. M. Odlyzko, editor, *Advances in Cryptology – Proceedings of CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa-Barbara, California, 1987. Springer-Verlag, Berlin.
28. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In B. Kaliski, editor, *Advances in Cryptology – proceedings of CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Santa-Barbara, California, 1997. Springer-Verlag, Berlin.
29. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In H. Imai and Y. Zheng, editors, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '99)*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Kamakura, Japan, March 1999. Springer-Verlag, Berlin.
30. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In M. Wiener, editor, *Advances in Cryptology – proceedings of CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa-Barbara, California, 1999. Springer-Verlag, Berlin.
31. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In J. Kilian, editor, *Advances in Cryptology – proceedings of CRYPTO '2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274, Santa-Barbara, California, 2001. Springer-Verlag, Berlin.
32. O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.
33. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
34. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of the 17th ACM Symposium on the Theory of Computing (STOC '85)*, pages 291–304, Providence, Rhode Island, U.S.A., 1985. ACM Press, New York.
35. S. Goldwasser, S. Micali, and R. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proceedings of the 25th Symposium on the Foundations of Computer Science (FOCS '84)*, pages 441–448, Singer Island, U.S.A., 1984. IEEE, New York.
36. 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.
37. C. Hall, I. Goldberg, and B. Schneier. Reaction Attacks Against Several Public-Key Cryptosystems. In *Proceedings of the International Conference on Information and Communications Security 1999*, Lecture Notes in Computer Science, pages 2–12. Springer-Verlag, 1999.
38. J. Håstad. Solving Simultaneous Modular Equations of Low Degree. *SIAM Journal of Computing*, 17:336–341, 1988.
39. A. Joux and R. Lercier. Improvements to the general Number Field Sieve for discrete logarithms in prime fields. *Mathematics of Computation*, 2000. to appear.

40. M. Joye, J. J. Quisquater, and M. Yung. On the Power of Misbehaving Adversaries and Security Analysis of the Original EPOC. In D. Naccache, editor, *The Cryptographers' Track at RSA Conference '2001 (RSA '2001)*, volume 2020 of *Lectures Notes in Computer Science*, pages 208–222, San Francisco , California, USA, April 2001. Springer-Verlag, Berlin.
41. 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/`.
42. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology – proceedings of CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa-Barbara, California, 1996. Springer-Verlag, Berlin.
43. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – proceedings of CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa-Barbara, California, 1999. Springer-Verlag, Berlin.
44. A. Lenstra and H. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.
45. A. Lenstra and E. Verheul. Selecting Cryptographic Key Sizes. In H. Imai and Y. Zheng, editors, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465, Melbourne, Australia, January 2000. Springer-Verlag, Berlin.
46. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
47. H.W. Lenstra. On the Chor-Rivest Knapsack Cryptosystem. *Journal of Cryptology*, 3:149–155, 1991.
48. J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In J. Kilian, editor, *Advances in Cryptology – proceedings of CRYPTO '2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238, Santa-Barbara, California, 2001. Springer-Verlag, Berlin.
49. R. J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN progress report*, 42-44:114–116, 1978. Jet Propulsion Laboratories, CALTECH.
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)*, volume 1556 of *Lectures Notes in Computer Science*, pages 72–80, Kingston, Ontario, Canada, 1998. Springer-Verlag, Berlin.
52. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing (STOC '90)*, pages 427–437, Baltimore, Maryland, U.S.A., 1990. ACM Press, New York.
53. V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
54. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
55. NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
56. NIST. Secure Hash Algorithm 256/384/512. October 2000.
57. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In H. Krawczyk, editor, *Advances in Cryptology – proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369, Santa-Barbara, California, 1998. Springer-Verlag, Berlin.
58. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In D. Naccache, editor, *The Cryptographers' Track at RSA Conference '2001 (RSA '2001)*, volume 2020 of *Lectures Notes in Computer Science*, pages 159–175, San Francisco , California, USA, April 2001. Springer-Verlag, Berlin.
59. T. Okamoto and D. Pointcheval. RSA–REACT: An Alternative to RSA–OAEP, September 2001. Second NESSIE Workshop.
60. T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In K. Kim, editor, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '2001)*, volume 1992 of *Lecture Notes in Computer Science*, Cheju Island, South Korea, February 2001. Springer-Verlag, Berlin.
61. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In H. Imai and Y. Zheng, editors, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '2000)*,

volume 1751 of *Lecture Notes in Computer Science*, pages 129–146, Melbourne, Australia, January 2000. Springer-Verlag, Berlin.

62. D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In U. Maurer, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, 1996. Springer-Verlag, Berlin.
63. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
64. D. Pointcheval and S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical report, Laboratoire d'Informatique de l'École Normale Supérieure, October 1996.
65. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.
66. M. O. Rabin. Digitalized Signatures. In R. Lipton and R. De Millo, editors, *Foundations of Secure Computation*, pages 155–166. Academic Press, New York, 1978.
67. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *Advances in Cryptology – Proceedings of CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Santa-Barbara, California, 1992. Springer-Verlag, Berlin.
68. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992.
69. 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.
70. RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.
Available from `http://www.rsa.com/rsalabs/pubs/PKCS/`.
71. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *Advances in Cryptology – Proceedings of CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 235–251, Santa-Barbara, California, 1990. Springer-Verlag, Berlin.
72. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
73. C. P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In T. Okamoto, editor, *Advances in Cryptology – Proceedings of ASIACRYPT '2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 458–469, Kyoto, 2000. Springer-Verlag, Berlin.
74. 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.
75. C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
76. H. Shimizu. On the Improvement of the Håstad Bound. In *1996 IEICE Fall Conference*, Volume A-162, 1996. In Japanese.
77. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *Advances in Cryptology – Proceedings of EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, 1997. Springer-Verlag, Berlin.
78. V. Shoup. OAEP Reconsidered. In J. Kilian, editor, *Advances in Cryptology – proceedings of CRYPTO '2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259, Santa-Barbara, California, 2001. Springer-Verlag, Berlin.
79. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In H. Imai and Y. Zheng, editors, *Workshop on Practice and Theory in Public-Key Cryptography (PKC '98)*, Lecture Notes in Computer Science, Yokohama, Japan, February 1998. Springer-Verlag, Berlin.
80. S. Vaudenay. Cryptanalysis of the Chor-Rivest Scheme. In H. Krawczyk, editor, *Advances in Cryptology – proceedings of CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 243–256, Santa-Barbara, California, 1998. Springer-Verlag, Berlin.
81. 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.