Proceedings of the 4th International Conference on Information Security and Cryptology '01
(6–7 december 2001, Seoul, South Korea)
K. Kim Ed., LNCS 2288, pages 1–17, Springer-Verlag 2002.

Practical Security in Public-Key Cryptography

David Pointcheval

LIENS – CNRS, École Normale Supérieure, 45 rue d'Ulm, F – 75230 Paris Cedex 05.
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. Moreover, for a long time the security proofs have only been performed in an asymptotic framework, which provides some confidence in the scheme but for very huge parameters only, and thus for unpractical schemes.
A recent trend consists in providing very efficient reductions, with a practical meaning: with usual parameters (such as 1024-bit RSA moduli) the computational cost of any attack is actually 2^{72}, given the state of the art about classical problems (*e.g.* integer factoring).
In this paper, we focus on practical schemes 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.

1 Introduction

1.1 Motivation

Since the beginning of public-key cryptography, with the seminal Diffie-Hellman paper [14], many suitable algorithmic problems for cryptography have been proposed (*e.g.* one-way —possibly trapdoor— functions). Then, many cryptographic schemes have been designed, together with more or less heuristic proofs of their security relative to the intractability of these problems. 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 Chor-Rivest cryptosystem [10] illustrates this fact quite well. This scheme based on the knapsack problem took more than 10 years to be totally broken [42] whereas before the effective attack it was believed to be very hard since all the classical techniques against the knapsack problems, such as LLL [26], had failed because of the high density of the involved instances. Therefore, the lack of attacks at some time should never be considered as a security validation of any 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 [3]): the proofs provide reductions from a well-studied problem 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, and thus they were trying to minimize the required assumptions on the primitives (one-way functions or permutations, possibly trapdoor, etc). 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 of proposed schemes. 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 have been expected, under the denominations of either “exact security” [7] or “concrete security” [31], 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, or relies on a weaker computational problem. Therefore, a classical way to give some convincing evidences about the security of an efficient scheme relative to a strong computational problem is to make some hypotheses on the adversary’s behavior: the attack is generic, independent of the actual implementation of some objects:

– of the hash function, in the “random oracle model” [17, 5];
– of the group, in the “generic (group) model” [28, 39].

1.3 Organization of the Paper

In the next 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. Such a formalism is the first step towards provable security. In section 3, we present some classical assumptions on which the security may rely. In sections 4 and 5, we describe several signature and encryption schemes with their formal security results and some detailed security proofs.

2 A First Formalism

2.1 Digital Signature Schemes

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

- The *key generation algorithm* K. On input 1^k, 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 *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 $\mathsf{k_p}$.

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 [24]:

- 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.*

On the other hand, various means can be made available to the adversary, helping her into the 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 amount of time is small, where

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

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.

2.2 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, 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. In this latter case, we can write $E(\mathsf{k_p}, m; r)$ where r is the random tape.
- 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 message space $\mathbf{M}$ and the internal random coins of the adversary and the encryption algorithm:

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

However, many applications require more, namely the *semantic security* (IND), *a.k.a. polynomial security/indistinguishability of encryptions* [22]. This security notion means 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 $\mathsf{Adv}^{\mathsf{ind}}(\mathbf{A})$, formally defined as

$$2 \times \Pr_b \begin{bmatrix} (\mathsf{k_p}, \mathsf{k_s}) \leftarrow K(1^k), (m_0, m_1, s) \leftarrow \mathbf{A}_1(\mathsf{k_p}), \\ c = E(\mathsf{k_p}, m_b) : \mathbf{A}_2(m_0, m_1, s, c) = b \end{bmatrix} - 1,$$

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) [15]. To break it, given a ciphertext, the adversary 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 [4] (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) [32]; a validity-checking oracle which, on input a ciphertext c, just answers whether it is a valid ciphertext or not (the so-called *reaction attacks* [21, 8]); or the decryption oracle itself, which on any ciphertext, except the challenge ciphertext, answers the corresponding plaintext (*non-adaptive [27]/adaptive [36] 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).

A general study of these security notions and attacks was conducted in [4]. We refer the reader to this paper for more details.

3 The Basic Assumptions

3.1 Computational Assumptions

For asymmetric cryptography, no security can be unconditionally guaranteed. Therefore, for any cryptographic protocol, security relies on a computational assumption: the existence of one-way functions, or permutations, possibly trapdoor.

Integer Factoring. 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. Unfortunately, it just provides a one-way function, without any possibility to invert the process. In 1978, Rivest, Shamir and Adleman [37] defined the so-called *RSA problem*: Let $N = pq$ be the product of two large primes of similar sizes and e an integer relatively prime to $\varphi(N)$. For a given $y \in \mathbb{Z}_N^\star$, find $x \in \mathbb{Z}_N^\star$ such that $x^e = y \bmod N$. The RSA assumption then says that this problem is intractable for any modulus $N = pq$, large enough (presumably as hard as factoring the modulus): the success probability $\mathsf{Succ}^{\mathsf{rsa}}(\mathbf{A})$ of any adversary $\mathbf{A}$ within a reasonable running time is small.

Discrete Logarithm. Some other classical problems are related to the discrete logarithm. The setting is quite general: one is given a finite cyclic group $\mathcal{G}$ of prime order q (such as a subgroup of $(\mathbb{Z}_p^\star, \times)$ for $q \mid p-1$, or an elliptic curve, etc) and a generator $\mathbf{g}$ (*i.e.* $\mathcal{G} = \langle \mathbf{g} \rangle$). In such a group, one considers the following problems (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 $\mathbf{c} = \mathbf{DH}(\mathbf{a}, \mathbf{b})$.

It is clear that they are sorted from the strongest problem to the weakest one. Very recently, Okamoto and the author [33] 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.

3.2 Ideal Objects

As already remarked, one often has to make some assumptions about the adversary's behavior. Let us present two classical models.

The Generic Model. Generic algorithms [28, 39], as introduced by Nechaev and Shoup, encompass group algorithms that do not exploit any special property of the encodings of group elements other than the property that each group element is encoded by a unique string. Remark that such algorithms are the only known for well-chosen elliptic curves. However, it is a strong and non-realistic restriction when one works in a subgroup of $\mathbb{Z}_p^\star$.

A *generic* algorithm $\mathbf{A}$ over a group $\mathcal{G}$ is a probabilistic algorithm that takes as input an *encoding list* $\{\sigma(x_1), \cdots, \sigma(x_k)\}$, where each x_i is in $\mathcal{G}$. An encoding of a standard group $\mathcal{G}$ is an injective map from $\mathcal{G}$ into a set of bit-strings S. While it executes, the algorithm may consult an oracle for further encodings. Oracle calls consist of triples $\{i, j, \epsilon\}$, where i and j are indices of the encoding list and ϵ is $\pm$. The oracle returns the string $\sigma(x_i \pm x_j)$, according to the value of ϵ and this bit-string is appended to the list, unless it was already present.

An interesting result in this model is the complexity lower-bound for breaking the **DL** problem. Similar results have been proven for all the above Diffie–Hellman problems [39, 1]. The consequence is that all these problems (**DL**, **CDH**, **DDH** and **GDH**) require an expected time in the square root of the order q to be solved by any generic algorithm.

Theorem 1. *Let $\mathcal{G}$ be a standard cyclic group of prime order q. Let $\mathbf{A}$ be a generic algorithm over $\mathcal{G}$ that makes at most n queries to the group-oracle. If $x \in \mathcal{G}$ and an encoding σ are chosen at random, then the probability that $\mathbf{A}$ returns x on input $\{\sigma(1), \sigma(x)\}$ is less than $1/q + (n+2)^2/q$.*

Proof. The idea of the proof is to identify the probabilistic space consisting of σ and x with the space $S^{n+2} \times \mathcal{G}$, where S is the set of bit-string encodings. Given a tuple $\{z_1, \cdots, z_{n+2}, x\}$ in this space, z_1 and z_2 are used as $\sigma(1)$ and $\sigma(x)$, the successive z_i are used in sequence to answer the oracle queries, modeled by formal linear relations of 1 and x, *i.e.*, linear polynomials $P_i = a_i + b_i X$. This interpretation may yield inconsistencies as it does not take care of possible collisions between oracle queries when evaluating the polynomials P_i in x, but with probability less than $(n+2)^2/q$. Eventually, let us note that the output of a computation corresponding to a good sequence $\{z_1, \cdots, z_{n+2}, x\}$ (which does not make two polynomials to collude in x) does not depend on x. $\square$

The Random Oracle Model. The "random oracle model" was the first to be introduced in the cryptographic community [17, 5]: the hash function is 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 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.

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.

4 Provably Secure Digital Signature Schemes

4.1 Basic Signature Schemes

The Plain-RSA Signature. Two years after the Diffie-Hellman paper [14], Rivest, Shamir and Adleman [37] proposed the first signature scheme based on the "trapdoor one-way permutation paradigm", using the RSA function: the key 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 $\mathbb{Z}_N$, 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 El Gamal Signature Scheme. In 1985, El Gamal proposed the first digital signature scheme based on the **DL** problem [16], but with no formal security analysis: the key generation algorithm produces a large prime p, as well as an element g in $\mathbb{Z}_p^\star$ of large order. It also creates a pair of keys, the private key $x \in \mathbb{Z}_{p-1}^\star$ and the public key $y = g^x \bmod p$. The signature of a message m is a pair (r, s), where $r = g^k \bmod p$, with a random $k \in \mathbb{Z}_{p-1}^\star$, and $s = (m - xr)/k \bmod p - 1$. This pair satisfies $g^m = y^r r^s \bmod p$, which is checked by the verification algorithm. Unfortunately, as above, existential forgeries are easy.

4.2 DL-Based Signatures

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 [23], and uses hash functions in order to create a kind of virtual verifier. The first application was derived from the Fiat–Shamir identification scheme [17]. This paradigm has also been applied by Schnorr [38], and provided the most efficient El Gamal-like scheme, with no easy existential forgery.

The security results for that paradigm have been considered as folklore for a long time but without any formal validation. However, Stern and the author [35] 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* [35]. However, for the Schnorr's signature scheme, one can just formally prove 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 $207 q_h T$. Actually, this security result is not practical, since q_h may be huge.

This technique has been applied on several other variants of the El Gamal [16] signature scheme, such as the Korean Standard KCDSA [25]. However, the American Standard DSA [29] does not fit with any of these designs. Therefore, this widely used scheme never got any formal security proof (even with a costly reduction). Recently, Brown [9] considered this standard in the generic model, which provides a practical result, under the assumption of generic adversaries. However, this makes this result possibly suitable for ECDSA only [2].

Description of ECDSA. The key generation algorithm defines an elliptic curve, and a point $\mathbf{g}$ of large prime order q. It also creates a pair of keys, the private key $x \in \mathbb{Z}_q$ and the public key $\mathbf{y} = x \cdot \mathbf{g}$. The signature of a message m is a pair (r, s): $\mathbf{r} = k \cdot \mathbf{g}$, with a random $k \in \mathbb{Z}_q^\star$, $r = x_\mathbf{r} \bmod q$, where $x_\mathbf{r}$ is the x-coordinate of $\mathbf{r}$, $e = H(m)$ and $s = k^{-1}(e + xr) \bmod q$. This pair satisfies $r = x_{\mathbf{r}'} \bmod q$ where $\mathbf{r}' = es^{-1} \cdot \mathbf{g} + rs^{-1} \cdot \mathbf{y}$, with $e = H(m)$, which is checked by the verification algorithm. It involves a hash function H which outputs h-bit long digests.

Theorem 2. *Let $\mathcal{G}$ be a standard cyclic group of prime order q. Let S be a set of bit-string encodings. Let $\mathbf{A}$ be a generic algorithm over $\mathcal{G}$ that makes at most q_s queries to the signing oracle and n queries to the group-oracle, with a running time bounded by t.*

If $\mathbf{A}$ can perform an existential forgery with probability greater than ε, for random x and random encoding σ, on input $\{\mathbf{g} = \sigma(1), \mathbf{y} = \sigma(x)\}$, then one can extract a collision for H with probability $\varepsilon' \geq \varepsilon - (n + q_s + 2)(n + 2)/q$, within almost the same time.

Proof. The proof uses the same technique as for the theorem 1. Let $\mathbf{A}$ be a generic attacker able to forge some message M with a signature (r, s). We describe several games, which differ just a little bit between each other [40].

Game_0: This is the game the generic adversary plays, with a random encoding σ, and a random pair of keys. The adversary eventually outputs a message M and a signature (r, s). We denote by S_0 the event $V(\mathsf{k_p}, M, (r, s)) = 1$ (as well as S_i in any Game_i below.) By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1: In this game, we simulate the encoding and the group-oracle using a random sequence $\{z_1, \cdots, z_{n+2}, x\}$, modeling oracle queries by linear polynomials $P_i = a_i + b_i X$: $|\Pr[S_1] - \Pr[S_0]| \leq (n + 2)^2/q$.

Game_2: We modify the random choice of the encoding oracle answers z_i. For all the queries $\sigma(b_\ell X + a_\ell)$, one gets a random $e_\ell \in_R \mathbb{Z}_q$, as well as a random $z_\ell \in S$ such that the x-coordinate of the corresponding point is equal to $b_\ell a_\ell^{-1} e_\ell \bmod q$. For convenient compact encoding sets, this simulation can be perfect: $\Pr[S_2] = \Pr[S_1]$.

Game_3: We modify the random choice of the e_ℓ when this latter is smaller than 2^h, where h is the bit-length of H-output: one gets a random message M_ℓ and computes $e_\ell = H(M_\ell) \bmod q$. Under the assumption of the uniformity of the output of H, which is related to the collision-resistance, $\Pr[S_3] = \Pr[S_2]$.

Game$_4$: In this game, we simulate the signing oracle, which can be perfectly performed by defining some values of the encoding, unless they have already been defined before: $|\Pr[S_4] - \Pr[S_3]| \leq nq_s/q$.

In this latter game, one can easily see that an existential forgery $(M, (r, s))$ leads to a collision for H, between M and some M_ℓ, if the bit-size of q is larger than h: $r = \sigma(H(M)s^{-1} + xrs^{-1}) = (rs^{-1}) \,/\, (H(M)s^{-1}) \times H(M_\ell) \bmod q$. □

Therefore, under the collision-resistance of H, implemented by SHA-1 [30], and $q > 2^{160}$, one gets a very tight security result against generic adversaries. However, this strong generic model is not as convincing as the random oracle model. Studies on elliptic curves may reveal non-generic attacks.

4.3 RSA-Based Signatures

In 1996, Bellare and Rogaway [7] proposed some signature schemes, based on the RSA assumption, provably secure in the random oracle model. The first scheme is the by now classical hash-and-decrypt paradigm (*a.k.a.* the Full-Domain Hash paradigm): instead of directly signing m using the RSA function, one first hashes it using a full-domain hash function $H : \{0,1\}^\star \to \mathbb{Z}_N$, and computes the e^{th} root, $\sigma = H(m)^d \bmod N$. Everything else is straightforward. For this scheme, named FDH-RSA, one can prove in the random oracle model [7, 11, 5]: for any adversary, her probability for 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, is upper-bounded by $3q_s\mathsf{Succ}^{\mathsf{rsa}}(t + (q_s + q_h)T_{exp})$, where T_{exp} is the time for an exponentiation to the power e, modulo N. This is quite bad because of the factor q_s. This factor is better than the factor q_h, as it was in the original proof [7], and for the **DL**-based signature schemes, but it is still too bad for practical security. Therefore, Bellare and Rogaway proposed a better candidate, the Probabilistic Signature Scheme (PSS): the key generation is still the same, but the signature process involves three hash functions

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

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, $\sigma = y^d \bmod N$.

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 RSA–PSS, Bellare and Rogaway proved the security in the random oracle model.

Theorem 3. *Let* $\mathbf{A}$ *be a CMA-adversary against RSA–PSS. Let us consider any adversary* $\mathbf{A}$ *which produces an existential forgery within a time* t*, after* q_F*,* q_G*,*

q_H and q_s queries to the hash functions F, G and H and the signing oracle respectively. Then her success probability is upper-bounded by

$$\mathsf{Succ}^{\mathsf{rsa}}(t+(q_s+q_H)k_2\cdot T_{exp}(k))+\frac{1}{2^{k_2}}+(q_s+q_H)\cdot\left(\frac{q_s}{2^{k_1}}+\frac{q_F+q_G+q_H+q_s+1}{2^{k_2}}\right),$$

with $T_{exp}(k)$ the time for an exponentiation modulo a k-bit integer.

Proof. First, we assume the existence of an adversary $\mathbf{A}$ that produces an existential forgery with probability ε within time t, after q_F, q_G and q_H queries to the random oracles F, G and H and q_s queries to the signing oracle. This is the game of the real-world attack (denoted below Game_0). In any Game_i, we denote by S_i the event $V(\mathsf{k_p}, m, \sigma) = 1$.

Game_1**:** In this game, we replace the random oracles F and G by random answers for any new query. This game is clearly identical to the previous one: $\Pr[S_1] = \Pr[S_0]$.

Game_2**:** Then, we replace the random oracle H by the following simulation. For any new query (m, r), one chooses a random $u \in \mathbb{Z}_N$ and computes $z = u^e \bmod N$, until the most significant bit of z is 0, but at most k_2 times (otherwise one aborts). Thereafter, z is parsed into $0 \,\|\, w \,\|\, s \,\|\, t$, and one defines $F(w) \leftarrow t$, $G(w) \leftarrow s \oplus r$ and $H(m, r) \leftarrow w$. Finally, one returns w. Let us remark that the number of calls to H is upper-bounded by q_s+q_H. This game may only differ from the previous one if during some H-simulations,
- z is still in the bad range, even after the k_2 attempts;
- $F(w)$ or $G(w)$ have already been defined.

$$|\Pr[S_2] - \Pr[S_1]| \le (q_H + q_s) \times \left(\frac{1}{2^{k_2}} + \frac{q_F + q_G + q_H + q_s}{2^{k_2}}\right).$$

Game_3**:** Now, we simply abort if the signing oracle makes a $H(m, r)$-query for some (m, r) that has already been asked to H. Furthermore, for any new query (m, r) directly asked by the adversary, one computes $z = yu^e \bmod N$, instead of $z = u^e \bmod N$. The distribution of the z is exactly the same as before. Thus the above abortion makes the only difference, which gives $|\Pr[S_3] - \Pr[S_2]| \le q_s(q_H + q_s)/2^{k_1}$.

Game_4**:** In the last game, we replace the signing oracle by an easy simulation, returning the value u involved in the answer $H(m, r) = z = u^e \bmod N$. The simulation is perfect, then $\Pr[S_4] = \Pr[S_3]$.

The event S_4 means that, at the end of Game_4, the adversary outputs a valid message/signature (m, σ). But in this game, it is only possible either by chance, or by inverting RSA: $\Pr[S_4] \le \mathsf{Succ}^{\mathsf{rsa}}(t', k) + 2^{-k_2}$, where t' is the running time of the adversary and the simulations: $t' \le t + (q_s + q_H)k_2 \cdot T_{exp}(k)$. □

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 v2.0 standard for signature.

Recently, Cramer and Shoup [13] proposed the first efficient signature scheme with a security proof in the standard model, and thus no ideal assumption. However, the security relies on a stronger computational assumption, the intractability of the so-called *flexible RSA problem*: Let $N = pq$ be the product of two large primes of similar sizes. For a given $y \in \mathbb{Z}_N^\star$, find a prime exponent e and $x \in \mathbb{Z}_N^\star$ such that $x^e = y \bmod N$.

The key generation algorithm produces a large composite number $N = pq$, where p and q are strong primes ($p = 2p'+1$ and $q = 2q'+1$, with p' and q' some prime integers). It also generates two non-quadratic residues $h, x \in \mathbb{Z}_N^\star$, and an $(\ell+1)$-bit prime integer e'. For signing a message m, one chooses a random non-quadratic residue $y \in \mathbb{Z}_N^\star$ and an $(\ell+1)$-bit prime integer e. Then one computes $x' = (y')^{e'} h^{-H(m)} \bmod N$, and solves the equation $y^e = xh^{H(x')} \bmod N$ for the unknown y. The signature is the triple (e, y, y'), with e an odd $(\ell+1)$-bit integer, which satisfies $(y')^{e'} = x'h^{H(m)} \bmod N$ and $y^e = xh^{H(x')} \bmod N$. The verification algorithm simply checks the above properties for the triple.

Even if the security holds in the standard model, the reduction is quite expensive (at least quadratic in the number q_s of queries asked to the signing oracle) and furthermore it is not tight, since once again, a factor q_s appears between the success probability for solving the *flexible RSA problem* and for breaking the signature scheme. Therefore, this security does not mean anything for practical parameters.

5 Provably Secure Public-Key Encryption Schemes

5.1 Basic Encryption Schemes

The Plain-RSA Encryption. The RSA primitive [37] can also be used for encryption: the key 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 $\mathbb{Z}_N$, is $c = m^e \bmod N$. This ciphertext can be 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 $\mathsf{k_p}$ is always the same, and thus it is easy to check whether a ciphertext c really encrypts m, by re-encrypting this latter. 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 $c' = c$ or not to make the decision.

The El Gamal Encryption Scheme. In 1985, El Gamal [16] also designed a **DL**-based public-key encryption scheme, inspired by the Diffie-Hellman key exchange protocol [14]: given a cyclic group $\mathcal{G}$ of prime order 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 $(\mathbf{c} = a \cdot \mathbf{g}, \mathbf{d} = a \cdot \mathbf{y} + \mathbf{m})$. 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 **CDH** problem. It is also IND-CPA relative to the **DDH** problem [41]. About OW-PCA, it relies on the new **GDH** problem [33]. However, it does not prevent adaptive chosen-ciphertext attacks because of the homomorphic property.

As we have seen above, the expected security level is IND-CCA. We wonder if we can achieve this strong security with practical encryption schemes.

5.2 The Optimal Asymmetric Encryption Padding

In 1994, Bellare and Rogaway proposed a generic conversion [6], in the random oracle model, the "Optimal Asymmetric Encryption Padding" (OAEP), which was claimed to apply to any family of trapdoor one-way permutations, such as RSA. The key generation produces a one-way permutation $f : \{0,1\}^k \rightarrow \{0,1\}^k$, the public key. The private key is the inverse permutation g, which requires a trapdoor to be actually computed. The scheme involves two hash functions

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

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. For that, one chooses a random string $r \in \{0,1\}^{k_0}$; 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, the trapdoor to compute 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 was the semantic security against non-adaptive chosen-ciphertext attacks (*a.k.a.* lunchtime attacks [27]). Recently, Shoup [40] 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 after an effective attack against the PKCS #1 v1.5 [8].

Fortunately, Fujisaki, Okamoto, Stern and the author [20] provided a complete security proof of IND-CCA-security for OAEP in general, but also for RSA–OAEP in particular under the RSA assumption.

The proof is a bit intricate, so we refer the reader to [20] for more information. However, our reduction is worse than the incomplete one originally proposed by Bellare and Rogaway [6]: an attacker in time t with advantage ε against RSA–OAEP can be used to break RSA with probability almost ε^2, but within a time bound $t + q_h^2 \times \mathcal{O}(k^3)$, where q_h is the total number of queries asked to the hash functions. Because of the quadratic term q_h^2, this reduction is meaningful for huge moduli only, more than 4096-bit long!

5.3 A Rapid Enhanced-security Asymmetric Cryptosystem Transform

Anyway, there is no hope to use OAEP with any **DL**-based primitive, even with huge parameters, because of the "permutation" requirement which limits the application of OAEP to RSA only. More general conversions have recently been proposed, first by Fujisaki and Okamoto [18, 19], then by the author [34], that apply to any OW-CPA scheme to make it into an IND-CCA one, still in the random oracle model. But the last one proposed by Okamoto and the author [32] is the most efficient: REACT (see figure 1). It applies to any encryption scheme $\mathbf{S} = (K, E, D)$

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

where $\mathbf{PK}$ and $\mathbf{SK}$ are the sets of the public and private keys, $\mathbf{M}$ is the message space, $\mathbf{C}$ is the ciphertext space and $\mathbf{R}$ is the random coin space. We also need two hash functions G and H,

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

where κ is the security parameter, while ℓ denotes the size of the messages to encrypt. About the converted scheme $\mathbf{S}' = (K', E', D')$, one can claim the

K': **Key Generation** $\to (\mathsf{k_p}, \mathsf{k_s})$
$(\mathsf{k_p}, \mathsf{k_s}) \leftarrow K(1^\kappa)$
E': **Encryption of** $m \in \mathbf{M}' = \{0,1\}^\ell \to (a, b, c)$
$R \in \mathbf{M}$ and $r \in \mathbf{R}$ are randomly chosen $a = E(\mathsf{k_p}, R; r) \qquad b = m \oplus G(R) \qquad c = H(R, m, a, b)$
D': **Decryption of** $(a, b, c) \to m$
Given $a \in \mathbf{C}$, $b \in \{0,1\}^\ell$ and $c \in \{0,1\}^\kappa$ $R = D(\mathsf{k_s}, a) \qquad m = b \oplus G(R)$ if $c = H(R, m, a, b)$ and $R \in \mathbf{M} \to m$ is the plaintext

Fig. 1. Rapid Enhanced-security Asymmetric Cryptosystem Transform $\mathbf{S}'$

following security result:

Theorem 4. *Let* $\mathbf{A}$ *be a CCA-adversary against the semantic security of* $\mathbf{S}'$. *If* $\mathbf{A}$ *can get 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 invert* E *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 time required by the PCA oracle to answer any query.*

Proof. We consider a sequence of games in which the adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ is involved. In each game, we use a random bit β and we are given $\alpha = E(\mathsf{k_p}, \rho; w)$, for random ρ, w. The adversary runs in two stages: given $\mathsf{k_p}$, $\mathbf{A}_1$ outputs a pair of messages (m_0, m_1), one encrypts $C' = (a', b', c') = E'(\mathsf{k_p}, m_\beta)$; on input C', $\mathbf{A}_2$ outputs a bit β'. In both stages, the adversary has access to the random oracles G and H, but also to the decryption oracle. In each game, we denote by S_i the event $\beta' = \beta$.

Game_0**:** This is the above real-world game: $\Pr[S_0] = (1+\varepsilon)/2$.

Game_1**:** In this game, we simulate the oracles G and H in a classical way, returning new random values for any new query: $\Pr[S_1] = \Pr[S_0]$.

Game_2**:** Then, we replace the decryption oracle by the following simulation. For each query (a, b, c), one looks at all the pairs (R, m) such that $H(R, m, a, b)$ has been asked. For any such R, one asks the Plaintext-Checking Oracle whether a is a ciphertext of R. Then it computes $K = G(R)$. If $b = K \oplus m$ then one outputs m as the plaintext of the triple (a, b, c). Otherwise, one rejects the ciphertext. One may remark that the probability of a wrong simulation is less than $1/2^{\kappa}$ (if the adversary guessed the H-value), therefore $|\Pr[S_2] - \Pr[S_1]| \leq q_D/2^{\kappa}$.

Game_3**:** Now, we modify the computation of C', given (m_0, m_1). Indeed, we set $a' \leftarrow \alpha$, and $b \in_R \{0,1\}^{\ell}$, $c \in_R \{0,1\}^{\kappa}$. Without asking $G(\rho)$ nor $H(\rho, m_i, a', b')$, the adversary cannot see the difference. In such a case we simply stop the game. Anyway, ρ would be in the list of queries asked to G or to H. It can be found after $q_G + q_H$ queries to the Plaintext-Checking Oracle: $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Succ}^{\mathsf{ow}}(\mathbf{A}')$, where $\mathbf{A}'$ is a PCA-adversary.

In this latter game, one can easily see that without having asked $G(\rho)$ or $H(\rho, m_i, a', b')$ to get any information about the encrypted message m, the advantage of the adversary is 0. This concludes the proof. □

Hybrid Cryptosystems. In this REACT conversion, one can improve efficiency, replacing the one-time pad by any symmetric encryption scheme, using $K = G(R)$ as a session key. Moreover, the symmetric encryption scheme is just required to be semantically secure under passive attacks, a very weak requirement. With RSA, but also any other deterministic primitive, the construction can be further improved, with just $c = H(R, m)$, or equivalently $c = H(R, b)$.

5.4 Practical Security

As for PSS only, but which was very specific to RSA, the security proof of REACT is both tight with a very efficient reduction in the widely admitted random oracle model: the cost of the reduction is linear in the number of oracle queries. Furthermore, the success probabilities are tightly related. Therefore, this scheme is perfectly equivalent to the difficulty of the underlying problem, without having to use larger parameters. For example, RSA–REACT with a 1024-bit modulus actually provides a provable security level in 2^{72}, whereas 1024-bit RSA–OAEP would provide a security level in 2^{36} only!

We cannot deal with provably secure encryption schemes without referring to the first efficient scheme proven in the standard model, proposed three years ago by Cramer and Shoup [12]. Actually, this encryption scheme achieves $\mathsf{IND\text{-}CCA}$, with a both tight and very efficient reduction, but to the **DDH** problem. Furthermore, the encryption and decryption processes are rather expensive (more than twice as much as other constructions in the random oracle model.)

6 Conclusion

In this paper, we reviewed several encryption and signature schemes with their security proofs. The security results are various, and have to be carefully considered since some of them are meaningless for usual sizes. Fortunately, several schemes have a practical significance. However, when one needs such a cryptographic scheme, one first has to decide between (unrelated) assumptions: a computational problem (*e.g.*, RSA, **CDH**, **GDH**) in the random oracle model; a decisional problem (*e.g.*, **DDH**) in the standard model; or the generic model. Second, the efficiency may also be a major criterion. Therefore security is still a matter of subtle trade-offs, until one finds a very efficient and secure scheme, relative to a strong problem in the standard model.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *CT – RSA '01*, LNCS 2020, pages 143–158. Springer-Verlag, Berlin, 2001.
2. American National Standards Institute. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm. ANSI X9.62-1998. January 1999.
3. M. Bellare. Practice-Oriented Provable Security. In *ISW '97*, LNCS 1396. Springer-Verlag, Berlin, 1997.
4. 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.
5. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
6. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
7. 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.
8. 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.
9. D. R. L. Brown. The Exact Security of ECDSA. January 2001.
   Available from `http://grouper.ieee.org/groups/1363/`.
10. 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.
11. J.-S. Coron. On the Exact Security of Full-Domain-Hash. In *Crypto '00*, LNCS 1880, pages 229–235. Springer-Verlag, Berlin, 2000.
12. 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.
13. 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.
14. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
15. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.

21. 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.
22. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
23. 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.
24. 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.
25. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. August 1998. Available from `http://grouper.ieee.org/groups/1363/`.
26. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
27. 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.
28. V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
29. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
30. NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
31. 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.
32. 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.
33. 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.
34. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
35. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
36. 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.
37. 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.
38. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
39. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266. Springer-Verlag, Berlin, 1997.
40. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.
41. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC '98*, LNCS. Springer-Verlag, Berlin, 1998.
42. S. Vaudenay. Cryptanalysis of the Chor-Rivest Scheme. In *Crypto '98*, LNCS 1462, pages 243–256. Springer-Verlag, Berlin, 1998.