Advances in Cryptology – Proceedings of Asiacrypt '2001 (9 – 13 december 2001, Gold Coast, Australia)
C. Boyd Ed. Springer-Verlag, LNCS 2248, pages 351–368.

Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks

Pierre-Alain Fouque and David Pointcheval

École Normale Supérieure, Département d'Informatique
45, rue d'Ulm, F-75230 Paris Cedex 05, France
{Pierre-Alain.Fouque,David.Pointcheval}@ens.fr

Abstract. Semantic security against chosen-ciphertext attacks (IND-CCA) is widely believed as the correct security level for public-key encryption scheme. On the other hand, it is often dangerous to give to only one people the power of decryption. Therefore, threshold cryptosystems aimed at distributing the decryption ability. However, only two efficient such schemes have been proposed so far for achieving IND-CCA. Both are El Gamal-like schemes and thus are based on the same intractability assumption, namely the Decisional Diffie-Hellman problem.
In this article we rehabilitate the twin-encryption paradigm proposed by Naor and Yung to present generic conversions from a large family of (threshold) IND-CPA scheme into a (threshold) IND-CCA one in the random oracle model. An efficient instantiation is also proposed, which is based on the Paillier cryptosystem. This new construction provides the first example of threshold cryptosystem secure against chosen-ciphertext attacks based on the factorization problem. Moreover, this construction provides a scheme where the "homomorphic properties" of the original scheme still hold. This is rather cumbersome because homomorphic cryptosystems are known to be malleable and therefore not to be CCA secure. However, we do not build a "homomorphic cryptosystem", but just keep the homomorphic properties.
Key words: Threshold Cryptosystems, Chosen-Ciphertext Attacks

1 Introduction

1.1 Chosen-Ciphertext Security

Semantic security against chosen-ciphertext attacks represents the correct security definition for a cryptosystem [31, 41, 4]. Therefore a lot of works [26, 25, 38, 34] have recently proposed schemes to convert any one-way function into a cryptosystem secure according to this security notion.

Before this notion, Naor and Yung in [33] proposed a weaker security notion that they called lunch-time attack (a.k.a. indifferent, or non-adaptive, chosen-ciphertext attack). The adversary can only ask decryption of ciphertexts before he receives the target ciphertext. Naor and Yung [33] presented a conversion to secure schemes against chosen-ciphertext attack in a lunch-time scenario. They used non-interactive zero-knowledge proof systems (proofs of membership [9, 8]) to show the consistency of the ciphertext, but not to prove that the people who built the ciphertext necessarily "knew its decryption".

Later Rackoff and Simon [41] refined this construction replacing the non-interactive zero-knowledge proofs of membership by non-interactive zero-knowledge proofs of knowledge. Therefore, when encrypting a message, one furthermore appends a non-interactive proof of knowledge of the plaintext, which leads to (adaptive) chosen-ciphertext secure cryptosystems. Indeed, the sender proves that he knows the plaintext and thus CCA is reduced to CPA.

© IACR 2001.

A similar notion has thereafter been defined, the so-called "plaintext-awareness" [7, 4], which means that when someone builds a valid ciphertext, he necessarily "knows" the corresponding plaintext. Therefore, a decryption oracle is unuseful for an adversary. But this latter notion is meaningful only in the random oracle model [6].

For few years, several efficient schemes have been proposed which achieve this high security level. Most of them have only been proven in the random oracle model [7, 27, 48, 36, 25, 26, 38, 34] using the plaintext-awareness property, but only one in the standard model [14].

1.2 Threshold Cryptosystems

On the one hand, in public-key cryptography in general, the ability of decrypting or signing is restricted to the owner of the secret key. This means that only one people has all the power. Whereas in some situations, such an ability should not be given to only one people, but shared among a group of users, such that a minimal number of them, the threshold, is needed to sign or decrypt.

On the other hand, the goal of cryptography is to withstand attackers. In the case of break-ins, *i.e.* adversary that can enter into a computer and steal the secret key, public-key systems in general are not protected against exposure of the secret key. As this kind of attacks done by intruders (hackers, Trojan horses) or by corrupted insiders are very common and frequently easy to perform, systems must be protected against them. Threshold cryptography can solve this problem by distributing trust among several components or servers. The secret key is then split into shares and each share is given to one of a group of servers.

First, the key generation process has to be distributed, in order to generate the shares of each server, without trusted party. This has been done in both the discrete logarithm [37, 30, 21], and the RSA [10, 24, 20] settings. For signature schemes, the signing process has been distributed in both environments [43, 29, 28, 22, 40, 47] as well.

For distributing the decryption process, similar techniques can be used, until one just wants to prevent chosen-plaintext attacks from passive adversaries (see below for precise definitions). However, when we want to prevent chosen-ciphertext attacks, in general, servers cannot start decryption before knowing whether the ciphertext is valid or not because an attacker can be one of these servers and in case of invalid ciphertexts, he had learned some information.

Consequently, when we try to share a cryptosystem, we should not wait until the end of the decryption to know whether the servers can really decrypt or not. Therefore, we have to integrate some proof of validity of the ciphertext that should be publicly verifiable. Unfortunately, most of all the known cryptosystems secure against chosen-ciphertext attacks are not suitable. Indeed, in the decryption processes, the alleged plaintext is decrypted, and the redundancy is checked just before returning the plaintext. Since the redundancy involves a hash function, the final check cannot be done efficiently in a distributed way.

1.3 Related Work

There are two methods to distribute the decryption process of a cryptosystem. Whereas the first one uses randomness, the second follows the model described by Lee and Lim in [32] where the usual decryption process for attaining cryptosystems immune against CCA is reversed: the receiver starts checking whether the ciphertext is valid before decrypting.

The first method has been proposed by Canetti and Goldwasser in [12]. In the Cramer-Shoup cryptosystem [14], the receiver can check the validity of a ciphertext by using one part of the secret key, before decrypting the valid ciphertext using the second part of the secret key. Therefore, one can think that it is easy to share this cryptosystem. Canetti and Goldwasser [12] succeeded in distributing this cryptosystem. But instead of checking the validity of the ciphertext in a first round and decrypting it according to the validity, they proposed a new strategy with only one round. The servers decrypt any ciphertext submitted and the decryption process is randomized. The servers compute $m \cdot (v'/v)^s$ where s is a random shared between the servers (part of the secret key), v the proof inside the ciphertext, and v' the proof calculated by the servers. In the centralized version, the decryption process verifies whether $v = v'$ or not. In the distributed version, if the proof is correct, $(v/v')^s = 1$ and the decryption gives m, otherwise it returns a random value. Nobody knows if the decrypted message is correct or not if there is no redundancy in the plaintext m. A solution is to decrypt twice the same ciphertext. If the results are the same, the message was well-formed. The main drawback is that the servers must keep in the secret key a sharing of a random s and hence, the length of the key is linear in the size of the number of decrypted messages. Consequently, even if the basic method with two rounds appears to be slower, it has nice features in term of storage and avoids the need of a protocol to compute a shared random.

This method is unfortunately specific to the Cramer-Shoup cryptosystem. The second method used by Shoup and Gennaro [48] follows Lee and Lim paper [32], with the El Gamal [17] cryptosystem, but in the random oracle model [6]. First, they tried to add a non-interactive zero-knowledge proof of knowledge of discrete logarithm, using the Schnorr signature [44]. But they remarked that the decryption simulation without the secret key would require an exponential time, because of a combinatorial explosion of the forking lemma [39]. This explosion can be avoided under stronger assumption [45]. They finally used non-interactive zero-knowledge proofs of membership (as in [33]) to avoid the rewinding, and thus the combinatorial explosion in the decryption simulation. In fact, the simulation of the decryption process cannot rewind the machine. The problem is the same as in the resettable zero-knowledge setting. Therefore, the same techniques of proof of membership in a hard language can be used [5]. We can note here that the proof of knowledge of Rackoff and Simon is actually a proof of membership. In this cryptosystem, there are two keys as in [33] : one which belongs to the receiver but the other one belongs to the sender. Since the prover has one of the two keys, he can decrypt and obtain the plaintext. Therefore, the proof turns to be a proof of knowledge for a specific sender. The sender can then decrypt messages and since it is a proof of membership we can simulate the proof without using rewinding technique.

1.4 The Basic Tool: Non-Interactive Zero-Knowledge Proof Systems

The model proposed by Naor and Yung strongly uses non-interactive zero-knowledge proofs of language membership in the common random string setting. Because of that, they had to restrict the power of the security model to lunch-time attacks since the adversary could use the target ciphertext and generate a new proof of membership. If the proof was correct, the decryption oracle decrypts it. But Naor and Yung cannot prove that the proof of membership cannot be changed by someone who does not know a witness. Indeed, they did not use any non-malleable property for the non-interactive zero-knowledge proof. Recently, this property has been considered [42], but only for theoretical proof systems.

In this paper, we use the idealized assumption of the random oracle model [6], which assumes that some functions behave like truly random functions. This allows to build efficient non-interactive zero-knowledge proofs, without the common random string setting, which achieve a weaker notion than non-malleability, but strong enough for our purpose, the *simulation soundness* [42].

Simulation Soundness. Let us consider any language $\mathcal{L}$, and a non-interactive zero-knowledge proof system for $\mathcal{L}$. For any adversary $\mathcal{A}$, with access to a proof $p^\star$, for a word $x^\star$, in or out of $\mathcal{L}$, we consider her ability to forge a new proof p, for a word out of $\mathcal{L}$. Therefore, for any adversary $\mathcal{A}$, we consider

$$\mathsf{Succ}^{\mathsf{sim-nizk}}(\mathcal{A}) = \Pr[(x,p) \leftarrow \mathcal{A}(Q) \,|\, x \in \bar{\mathcal{L}} \wedge (x,p) \notin Q],$$

having access to a bounded list Q of proven words $(x^\star, p^\star)$, where the word $w^\star$ is any word (in or out of the language $\mathcal{L}$) and $p^\star$ an accepted proof for $w^\star$. We denote by $\bar{\mathcal{L}}$ the complement of $\mathcal{L}$, and thus all the words out of the language $\mathcal{L}$.

More generally, we denote by $\mathsf{Succ}^{\mathsf{sim-nizk}}(t)$ the maximal success probability over any adversary, with running time bounded by t, in forging a new accepted proof for an invalid word, even after having seen a bounded number of accepted proofs on (in)valid words. In our situation, this bounded number will just be one.

This is a stronger notion than the classical soundness for non-interactive zero-knowledge proofs, but a weaker than non-malleability. Indeed, Sahai [42] showed that non-malleability of non-interactive zero-knowledge proofs implies this notion, that he calls *simulation soundness.*

As we see in the sequel, in the random oracle model, we can provide efficient proofs which achieve this security level.

1.5 Our solution

Fujisaki and Okamoto [26] proposed a generic conversion from any $\mathsf{IND\text{-}CPA}$ cryptosystem into an $\mathsf{IND\text{-}CCA}$ one, in the random oracle model [6]. In this paper, we revisit the twin-encryption technique of Naor and Yung [33], by providing a generic conversion from any $\mathsf{IND\text{-}CPA}$ cryptosystem into an $\mathsf{IND\text{-}CCA}$ one with publicly verifiable validity of the ciphertext (in front of the same kind of adversary, see below). Namely, this conversion provides threshold cryptosystems strongly secure. We furthermore present practical instantiations in the random oracle model, which achieve $\mathsf{IND\text{-}CCA}$ against active and adaptive adversaries.

2 Security Model

2.1 The Network

We assume a group of ℓ (probabilistic) servers, all connected to a common broadcast medium, called the communication channel. It can be an asynchronous channel like the Internet.

2.2 The Adversary

The adversary is computationally bounded and it can corrupt servers at any time by viewing the memories of corrupted servers (passive adversary), and/or modifying their behavior (active adversary). The adversary decides on whom to corrupt at the start of the protocol (static adversary). We also assume that the adversary corrupts no more than t out of ℓ servers throughout the protocol, where $\ell \geq 2t + 1$.

2.3 Threshold Cryptosystems

A t out of ℓ threshold cryptosystem consists of the following components:

- A *key generation algorithm* $\mathcal{K}$ that takes as input a security parameter in unary notation 1^k, the number ℓ of decryption servers, and the threshold parameter t; it outputs a *public key* pk, a list $\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell$ of private keys (which represents a sharing of the private key sk) and a list $\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell$ of verification keys.
- An *encryption algorithm* $\mathcal{E}$ that takes as input the public key pk and a cleartext m, and outputs a ciphertext c.
- Several *decryption algorithms* $\mathcal{D}_i$ (for $1 \leq i \leq \ell$) that take as input the public key pk, the private key sk_i, a ciphertext c, and output a *decryption share* σ_i (which may include a verification part to achieve robustness).
- A *recovery algorithm* that takes as input the public key pk, a ciphertext c, and a list $\sigma_1, \ldots, \sigma_\ell$ of decryption shares (or at least $t+1$ of them), together with the verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell$, and outputs a cleartext m or rejects if less than $t+1$ decryption shares are correct in the case of active adversaries. All users can run this algorithm.

2.4 Security Notions

In this section, we define the game an adversary plays and tries to win in order to achieve the goal of the attack. Adversary against threshold cryptosystems tries to attack the two following properties :

- Security of the underlying primitive. In the case of cryptosystem, it means one-wayness, semantic security [31], or non-malleability [16].
- Robustness. This means that corrupted players should not be able to prevent uncorrupted servers from decrypting ciphertexts. This notion is useful only in the presence of active adversaries. In other terms, it means that the decryption service is available even if the adversary can send bad decryption shares.

A user who wants to decrypt a ciphertext c sends it to a special server, called the *combiner*, who forwards it to all servers. The servers start checking the validity of the ciphertext, then compute a decryption share σ_i and eventually return it to the combiner. This latter combines the decryption shares to obtain the plaintext m and returns it to the user. If we want to withstand active adversaries, the combiner must decide when he receives decryption shares σ_i whether they are valid or not. A nice way is to use checking protocols [23], and verification keys are consequently needed. The goal of checking protocols is to allow each server to prove to others that it has achieved its task correctly.

Semantic Security. In the following, we focus on the semantic security [31] goal, denoted IND, and forget any other security notions (one-wayness and non-malleability.) Therefore, the game to consider is the following :

1. The key generation algorithm $\mathcal{K}$ is run. The adversary therefore receives the public key pk. With this public key, the adversary has the ability to encrypt any plaintext of his choice (hence the basic "chosen-plaintext attack").
2. The adversary chooses two cleartexts m_0 and m_1. These are given to an "encryption oracle" that chooses $b \in \{0, 1\}$ at random, encrypts m_b and gives the ciphertext c to the adversary.
3. At the end of the game, the adversary outputs $b' \in \{0, 1\}$. We say that the adversary wins the game if $b' = b$.

Semantic security against chosen-plaintext attack means that for any polynomial time bounded adversary, $b' = b$ with probability only negligibly greater than 1/2.

Chosen Ciphertext Attacks. A stronger attack is usually considered, the so-called chosen-ciphertext attack [41], in which the adversary is given a full access to the decryption oracle $\mathcal{D}_{\mathsf{sk}}$, feeding it with any ciphertext. It therefore obtains the corresponding plaintext, or the "reject" answer. There is the trivial restriction not to ask the challenge ciphertext.

Threshold Security. The above attacks are the classical attacks in the standard (non-threshold) setting of the cryptosystem. Even if it is a threshold one, the view of the adversary is the same as if there would be only one secret key. However, in the threshold setting, we have to consider the leakage of decryption shares. To this aim, we give a new oracle access to the adversary: the adversary is given a full access to the decryption oracles $\mathcal{D}_{\mathsf{sk}_i}$, but feeding them with a valid pair of plaintext-ciphertext. It therefore obtains the decryption share σ_i. If the pair is not valid (the ciphertext does not encrypt the given plaintext) the oracle may output anything [19]. This is therefore the basic security notion (for both $\mathsf{IND\text{-}CPA}$ and $\mathsf{IND\text{-}CCA}$) in the threshold setting: $\mathsf{IND\text{-}TCPA}$ and $\mathsf{IND\text{-}TCCA}$ respectively.

As explained in the motivation of threshold cryptosystems, such a scheme should resist to the corruption of some servers. Therefore, we have to consider this situation, which means that the adversary has control of some servers:

- still playing honestly — the adversary is thus a **passive** adversary. He has access to any internal data of some servers, but cannot modify their behavior.
- or modifying their behavior — the adversary is then an **active** adversary.

To sum up, we have several possible mixes of attacks and adversaries: the chosen-plaintext (CPA) or chosen-ciphertext (CCA) attacks, performed by passive ($\mathsf{\text{-}Passive}$) or active ($\mathsf{\text{-}Active}$) adversaries. According to the choice of corrupted servers, we consider adaptive or non-adaptive adversaries. Non-adaptive adversaries make their choice first (before anything else), whereas adaptive ones make their choice along the attack, adaptively. It has been proven that passive and adaptive adversaries are equivalent to passive and non-adaptive adversaries, when the number of servers is logarithmic [11].

One may remark that in the particular case where $\ell = 1$ and $t = 0$, we are back to the classical situation, where passive/active and (non)-adaptive adversaries are meaningless.

3 Generic Conversions into IND-CCA Cryptosystems

In this section, we revisit the twin-encryption paradigm proposed by Naor and Yung [33], while assuming that $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a (possibly threshold) cryptosystem which already achieves semantic security against chosen-plaintext attacks ($\mathsf{IND\text{-}CPA}$ or $\mathsf{IND\text{-}TCPA}$, in the threshold setting). Then, we provide a new scheme which prevents CCA (or TCCA, resp.) whatever the kind of adversary.

3.1 Generic Conversion GC

The Key Generation: $\mathbf{K}(1^k)$ runs twice $\mathcal{K}(1^k)$ to get two public keys $(\mathsf{pk}, \mathsf{pk}')$, which represent the new public key $\mathbf{PK}$. The same way, one defines the new set of secret keys as $\mathbf{SK} = \{\mathbf{SK}_i\}_{1\le i\le \ell} = \{\mathsf{sk}, \mathsf{sk}'\} = \{\mathsf{sk}_i, \mathsf{sk}'_i\}_{1\le i\le \ell}$ and the new set of verification keys $\mathbf{VK} = \{\mathbf{VK}_i\}_{1\le i\le \ell} = \{\mathsf{vk}, \mathsf{vk}'\} = \{\mathsf{vk}_i, \mathsf{vk}'_i\}_{1\le i\le \ell}$.

Encryption of m

- one first encrypts twice m under pk and pk', $a_0 = \mathcal{E}_{\mathsf{pk}}(m)$ and $a_1 = \mathcal{E}_{\mathsf{pk}'}(m)$;
- one then builds a proof that both ciphertexts encrypt the same plaintext under the keys pk and pk' respectively, $c = \mathsf{Proof}[\mathsf{pk}, \mathsf{pk}', \mathcal{D}_{\mathsf{sk}}(a_0) = \mathcal{D}_{\mathsf{sk}'}(a_1)]$.

Partial Decryption of (a_0, a_1, c)

- the server checks the validity of the proof c;
- it computes both decryption shares of the ciphertexts a_0 and a_1 (only one could be enough, but the same random choice should be done by all the servers).

It is then possible to reconstruct the plaintext, using the recovery algorithm.

With this generic construction, it is not clear that the proof c does not leak any information (as remarked in [33]), furthermore such a proof can seldom be done efficiently in the standard model. However, the random oracle model allows to make efficient non-interactive zero-knowledge proofs [39].

3.2 Non-Interactive Zero-Knowledge Proofs

In order to make the following proof to work, we need a strong security notion about the proof c on the language

$$\mathcal{L} = \{(\mathsf{pk}, \mathsf{pk}', \mathcal{E}_{\mathsf{pk}}(m), \mathcal{E}_{\mathsf{pk}'}(m)) \mid \forall m\},$$

called *simulation soundness* [42].

Indeed, we want that any adversary $\mathcal{A}$, having seen a pair $(x^\star, c^\star)$, where $x^\star = (\mathsf{pk}, \mathsf{pk}', \mathcal{E}_{\mathsf{pk}}(m), \mathcal{E}_{\mathsf{pk}'}(m'))$ (with $m = m'$ but also possibly $m \neq m'$) and $c^\star$ an accepted proof for $x^\star$, has a negligible success probability in forging a new proof c for a word $x \notin \mathcal{L}$:

$$\mathsf{Succ}^{\mathsf{sim-nizk}}(\mathcal{A}) = \Pr[(x, c) \leftarrow \mathcal{A}(x^\star, c^\star) \mid x \in \bar{\mathcal{L}} \wedge (x, c) \neq (x^\star, c^\star)].$$

The idea behind this success probability is that the adversary should not be able to build a new proof from previous ones, excepted for valid words (which means in $\mathcal{L}$). Indeed, one cannot avoid the adversary to build an accepted proof for a correct word chosen by herself, and in such a case the ciphertext is valid.

Furthermore, the adversary has access to a proof for a word in $\mathcal{L}$, or maybe out of $\mathcal{L}$, because the simulator will sometimes create an accepted proof for a word that is not in $\mathcal{L}$. Such a proof should not give any further information to the adversary either.

The proof c convinces everybody that the ciphertext is valid before starting the decryption. In the security proof, the decryption simulator knows one secret key. But the challenge ciphertext will not necessarily be a valid one (possibly with two distinct encrypted messages). Thanks to the random oracle model, it is still possible to simulate, in an indistinguishable way, an accepted proof even for such a wrong string, under the assumption of the intractability of the problem of deciding membership (a weaker assumption than the semantic security of the underlying cryptosystem).

Finally, we present some practical non-interactive zero-knowledge proofs, which are easily proven to be simulation-sound using the forking lemma technique [39].

3.3 Security Proof

We show that from any adversary $\mathcal{A}$ against $\mathsf{IND\text{-}CCA}$ of twin scheme, we can build an adversary $\mathcal{B}$ against $\mathsf{IND\text{-}CPA}$ of the original scheme, first only considering passive adversaries.

3.4 Passive Adversaries

Theorem 1. *Given an IND-CPA (or IND-TCPA) cryptosystem $\mathcal{S}$, the twin conversion provides an IND-CCA (or IND-TCCA, resp.) cryptosystem $\mathcal{S}_{tw}$, in the random oracle model.*

Proof. Our proof proceeds by reduction. Given a (t, ε)-adversary $\mathcal{A}$ against our scheme $\mathcal{S}_{tw}$ in the sense of $\mathsf{IND\text{-}CCA}$, we build a (t', ε')-attacker $\mathcal{B}$ against scheme $\mathcal{S}$ where $t' = t$ and $\varepsilon' = (\varepsilon - 9 \cdot \mathsf{Succ}^{\mathsf{sim-nizk}}(t))/4$.

First of all, one can note that if a (classical) cryptosystem is $\mathsf{IND\text{-}CPA}$, then if we encrypt the same message under two different public keys, the resulting twin-cryptosystem is still $\mathsf{IND\text{-}CPA}$. This result can be shown by applying hybrid techniques [31] and it has already been formally proven in [3, 2], with a advantage loss (divided by 2).

Now, we show how to make the reduction. The attacker $\mathcal{B}$ receives a given public key pk and we show how this attacker can use the adversary $\mathcal{A}$ that breaks $\mathsf{IND\text{-}CCA}$ to win the game ($\mathsf{IND\text{-}CPA}$). The simulator $\mathcal{B}$ runs $\mathcal{K}(1^k)$ and gets $(\mathsf{pk}', \mathsf{sk}' = \{\mathsf{sk}'_i\})$. He tosses a coin b, and sets $\mathsf{pk}_b = \mathsf{pk}$, while $\mathsf{pk}_{1-b} = \mathsf{pk}'$. Then, he sends $(\mathsf{pk}_0, \mathsf{pk}_1)$ to $\mathcal{A}$.

At the step 2 in the game, the adversary $\mathcal{A}$ outputs two messages m_0, m_1. The simulator $\mathcal{B}$ sends them to the challenger: the challenger chooses at random a bit b' and encrypts $m_{b'}$ under $\mathcal{E}_{\mathsf{pk}_b}$, yielding to $a_b^\star = \mathcal{E}_{\mathsf{pk}_b}(m_{b'})$.

Then, $\mathcal{B}$ tosses a new coin b'' at random and computes $a_{1-b}^\star = \mathcal{E}_{\mathsf{pk}_{1-b}}(m_{b''})$ and sends to the adversary the target ciphertext $y^\star = (a_0^\star, a_1^\star, c^\star)$, where $c^\star$ is a simulated proof of correctness of $a_0^\star$ and $a_1^\star$, which can be done in an indistinguishable way in the random oracle model, under the intractability of the decision problem: do $a_0^\star$ and $a_1^\star$ encrypt the same message?

Now, we show how to simulate the decryption oracle. Adversary $\mathcal{A}$ can perform queries $y = (a_0, a_1, d)$ to the decryption oracle, at any time, where $a_i = \mathcal{E}_{\mathsf{pk}_i}(m)$ and c is a proof of correctness of the ciphertext. The simulator $\mathcal{B}$ easily decrypts a_{1-b}, as he knows the secret keys related to $\mathsf{pk}_{1-b} = \mathsf{pk}'$. If the proof is correct we know that a_0 and a_1 encrypt the same value m. This simulation is perfect. If the proof is not correct, but accepted, the adversary had broken the simulation soundness, after having seen only one proof.

Finally, $\mathcal{A}$ answers with a bit $b^\star$, which is output by $\mathcal{B}$. Since the simulation may not be perfect, the adversary may never stop. In this latter case, after a time-out, $\mathcal{B}$ flips a coin $b^\star$. This latter has won if $b^\star = b'$, and thus with probability

$$\frac{\varepsilon' + 1}{2} = \Pr[b^\star = b' \wedge \mathsf{NIZK}] + \Pr[b^\star = b' \wedge \neg\mathsf{NIZK}] \geq \Pr[b^\star = b' \mid \mathsf{NIZK}] \cdot \Pr[\mathsf{NIZK}].$$

In the above formula, NIZK denotes the event that none of the proofs sent by the adversary to the decryption oracle breaks the simulation soundness, after having possibly seen one proof.

Indeed, if the adversary can forge proofs of membership, for wrong words, the simulator will always answer with the message encrypted under pk'. Therefore, the adversary can decide which key has the simulator.

However, under the assumption NIZK, saying that the adversary did not forge a wrong proof, our simulation of the decryption oracle is perfect. Then, using the notation pr for probabilities under this assumption:

– in the case $b'' = b'$, the simulation is perfect. Indeed, the challenge ciphertext is a valid ciphertext, and all the decryption queries are valid ciphertexts (under the NIZK assumption). And thus, the advantage is greater than $\varepsilon/2$, thanks to results about multicast encryption [2, 3] (excepted a possible advantage in the real game thanks to an attack on the soundness). Thus

$$\mathsf{pr}[b^\star = b' \mid b'' = b'] \geq \frac{\varepsilon/2+1}{2} - \Pr[\neg\mathsf{NIZK}] = \frac{\varepsilon}{4} + \frac{1}{2} - \Pr[\neg\mathsf{NIZK}].$$

– in the case $b'' \neq b'$, even a powerful adversary that can decrypt a_0 and a_1, will obtain m_0 and m_1. Therefore, he cannot get any advantage. However, the adversary who detects it may choose to never stop, or to cheat. If she decides to never stop, the time-out makes $\mathcal{B}$ to flip a coin. If she tries to cheat, she has no information about b'. Then, $\mathsf{pr}[b^\star = b' \mid b'' \neq b'] = 1/2$.

Therefore,

$$\begin{aligned}\frac{\varepsilon'+1}{2} &\geq \left(\frac{\mathsf{pr}[b^\star = b' \mid b'' = b'] + \mathsf{pr}[b^\star = b' \mid b'' \neq b']}{2}\right) \cdot \Pr[\mathsf{NIZK}] \\ &\geq \frac{1}{2} \cdot \left(\frac{\varepsilon}{4} + 1 - \Pr[\neg\mathsf{NIZK}]\right) \cdot \Pr[\mathsf{NIZK}] \geq \frac{1}{2} \cdot \left(\frac{\varepsilon}{4} + 1 - \frac{9}{4} \cdot \Pr[\neg\mathsf{NIZK}]\right).\end{aligned}$$

And thus,

$$\varepsilon' = 2\Pr[b^\star = b'] - 1 \geq \frac{\varepsilon - 9 \cdot \Pr[\neg\mathsf{NIZK}]}{4}.$$

In order to upper bound $\Pr[\neg\mathsf{NIZK}]$, we play the same game but knowing the two secret keys. Then, as soon as the adversary produces an accepted proof for an invalid word, we detect it, and thus output it. This breaks the simulation soundness with time t: $\Pr[\neg\mathsf{NIZK}] \leq \mathsf{Succ}^{\mathsf{sim-nizk}}(t)$. □

3.5 Active Adversaries

It is clear that the proof still holds whatever the adversary is, even in the threshold setting. We provided a rigorous proof without any corruption. But if the underlying scheme already prevents $\mathsf{IND\text{-}TCPA}$ against passive or active adversaries, the new one even prevents $\mathsf{IND\text{-}TCCA}$ against the same kind of adversaries.

4 Examples

The first example of semantically secure cryptosystem with easy proofs of equality of plaintexts is certainly the El Gamal cryptosystem [17]. Even if more efficient threshold versions have already been proposed [48] (even in the standard model [12]), we apply the first conversion on it.

The second example will provide the first RSA-based threshold cryptosystem secure under chosen-ciphertext attacks, even against active and adaptive adversaries. It is based on the Paillier's cryptosystem [35, 19]. Another version to share Paillier cryptosystem appears in [15].

In this part, we describe the cryptosystems and we insist on the proofs of membership which are specific.

4.1 The El Gamal Cryptosystem

Description of the El Gamal Cryptosystem. Let p be a strong prime, such that $q|p-1$ is also a large prime, and g be an element of $\mathbb{Z}_p^*$ of order q. We thus denote by G the subgroup of $\mathbb{Z}_p^*$ of the elements of order q. It is spanned by g. Let $y = g^x$ be the public key corresponding to the secret key x. To encrypt a message $M \in G$, randomly choose $r \in \mathbb{Z}_q$ and compute the ciphertext $(M.y^r, g^r)$. To decrypt a ciphertext $a = (\alpha, \beta)$, the receiver computes α/β^x. It is well-known that the semantic security of El Gamal is based on the Decisional Diffie-Hellman (DDH) problem [49].

IND-CPA Threshold Version of El Gamal Cryptosystem. The secret key x is split with Shamir secret sharing scheme. Each server has a share sk_i of the secret key sk and a verification key $\mathsf{vk}_i = g^{\mathsf{sk}_i}$. To decrypt a ciphertext $a = (\alpha, \beta)$, each server computes a decryption share $\beta_i = \beta^{\mathsf{sk}_i}$, and proves that $\log_g \mathsf{vk}_i = \log_\beta \beta_i$. The combiner selects a set S of $t+1$ correct shares and computes

$$\beta^x = \prod_{i \in S} \beta_i^{\lambda_{0,i}^S} \bmod p$$

where $\lambda_{i,0}^S$ denote the symbol of Lagrange. Finally, the combiner computes $\alpha/\beta^x \bmod p$ to recover the plaintext. One can easily show that if an adversary can break the semantic security of this cryptosystem, one can build an attacker that can break the semantic security of El Gamal, and thus the DDH assumption.

IND-CCA Threshold Version of El Gamal Cryptosystem. We can therefore apply previous twin conversion. One still gets one group G, with a generator g of prime order. Then the key generation algorithm is run twice and the public keys are $y_0 = g^{x_0}$ and $y_1 = g^{x_1}$. To encrypt a message M, the sender computes $a_0 = (M \cdot y_0^r, g^r) = (\alpha_0, \beta_0)$ and $a_1 = (M \cdot y_1^s, g^s) = (\alpha_1, \beta_1)$.

The proof of equality of plaintexts consists in proving the existence of r and s such that $\beta_0 = g^r$, $\beta_1 = g^s$ and $\alpha_0/\alpha_1 = y_0^r y_1^{-s}$.

To this aim, one chooses random $a, b \in \mathbb{Z}_q$, and computes $A = g^a$, $B = g^b$ and $C = y_0^a y_1^b$. Then, one gets the random challenge $e \in \mathbb{Z}_q$ from a hash function which is assumed to behave like a random oracle: $e = H(g, y_0, y_1, a_0, a_1, A, B, C)$. Eventually, one computes $\rho = a - re \bmod q$ and $\sigma = b + se \bmod q$. This proof can be easily verified by $A = g^\rho \beta_0^e$, $B = g^\sigma \beta_1^{-e}$, and $C = y_0^\rho y_1^\sigma (\alpha_0/\alpha_1)^e$, or equivalently by

$$e = H(g, y_0, y_1, a_0, a_1, g^\rho \beta_0^e, g^\sigma \beta_1^{-e}, y_0^\rho y_1^\sigma (\alpha_0/\alpha_1)^e),$$

where the proof consists of the tuple (e, ρ, σ).

The decryption process is straightforward, using the same technique as presented above, but twice, after having checked the validity of the ciphertext.

Security Analysis. The basic threshold El Gamal cryptosystem is clearly $\mathsf{IND\text{-}CPA}$. The generic conversion makes then the new proposal to be $\mathsf{IND\text{-}TCCA}$, but under the condition that the above proof of equality of plaintexts is simulation-sound. We thus have to prove it.

First, we have to be able to build a list Q of accepted proofs for words in and out of the language. This can easily be done, thanks to the random oracle property of H: one chooses ρ, σ and e in $\mathbb{Z}_q$, and defines

$$H(g, y_0, y_1, a_0, a_1, g^\rho \beta_0^e, g^\sigma \beta_1^{-e}, y_0^\rho y_1^\sigma (\alpha_0/\alpha_1)^e) \leftarrow e.$$

Now, let us assume that with access to this list of proofs, an adversary is able to forge a new proof for a wrong word $(\mathsf{pk}_0, \mathsf{pk}_1, a_0, a_1)$, with probability ν, within time t. Since everything is included in the query to the random oracle H, we can apply the forking lemma [39], which claims that

Lemma 2. *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine which can ask q_h queries to the random oracle, with $q_h > 0$. We assume that, within the time bound t, $\mathcal{A}$ produces, with probability $\nu \geq 7q_h/q$, a new accepted proof for a wrong word $(\mathsf{pk}_0, \mathsf{pk}_1, a_0, a_1)$, $(g, y_0, y_1, a_0, a_1; A, B; e; \rho, \sigma)$. Then, within time $t' \leq 16 q_h t/\nu$, and with probability $\nu' \geq 1/9$, a replay of this machine outputs two accepted proofs of a wrong word $(\mathsf{pk}_0, \mathsf{pk}_1, a_0, a_1)$:*

$$(g, y_0, y_1, a_0, a_1; A, B; e_0; \rho_0, \sigma_0) \text{ and } (g, y_0, y_1, a_0, a_1; A, B; e_1; \rho_1, \sigma_1),$$

with $e_0 \neq e_1 \bmod q$.

Let us assume that the adversary has not broken the collision intractability of H, then

$$g^{\rho_0}\beta_0^{e_0} = g^{\rho_1}\beta_0^{e_1}, \quad g^{\sigma_0}\beta_1^{-e_0} = g^{\sigma_1}\beta_1^{-e_1}$$
$$y_0^{\rho_0} y_1^{\sigma_0} (\alpha_0/\alpha_1)^{e_0} = y_0^{\rho_1} y_1^{\sigma_1} (\alpha_0/\alpha_1)^{e_1}$$

and thus,

$$\beta_0 = g^\rho, \ \beta_1 = g^\sigma, \text{ and } \alpha_0/\alpha_1 = y_0^\rho y_1^{-\sigma},$$

where

$$\rho = \frac{\rho_1 - \rho_0}{e_0 - e_1} \bmod q, \text{ and } \sigma = \frac{\sigma_0 - \sigma_1}{e_0 - e_1} \bmod q.$$

Since $\alpha_0 = M_0 y_0^\rho$, and $\alpha_1 = M_1 y_1^\sigma$, we eventually get $M_0 = M_1$, which means that the word is in the language, unless one has broken the collision intractability for H. But under the random oracle assumption, to get a probability greater than 1/9 to find a collision, one has to have asked more than $\sqrt{q}/3$ queries to H, using the birthday paradox, and thus

$$\frac{16 q_h t}{\nu} \geq t' \geq \frac{\sqrt{q}}{3}\tau,$$

where τ is the time required for an evaluation of H. This leads to

$$\mathsf{Succ}^{\mathsf{sim-nizk}}(t) \leq \nu \leq 48 \frac{q_h}{\sqrt{q}} \frac{t}{\tau}.$$

This proves the soundness of the proof system. But since this lemma still holds, even for an adversary with auxiliary information (the list Q), it furthermore proves the simulation soundness.

4.2 The Paillier Cryptosystem

Review of the Basic Cryptosystem. The Paillier cryptosystem is based on the properties of the Carmichael lambda function in $\mathbb{Z}_{n^2}^*$. We recall here the main two properties: for any $w \in \mathbb{Z}_{n^2}^*$,

$$w^{\lambda(n)} = 1 \bmod n, \quad \text{and} \quad w^{n\lambda(n)} = 1 \bmod n^2$$

Let n be an RSA modulus $n = pq$, where p and q are prime integers. Let g be an integer of order $n\alpha$ modulo n^2. The public key is $\mathsf{pk} = (n, g)$ and the secret key is $\mathsf{sk} = \lambda(n)$. To encrypt a message $M \in \mathbb{Z}_n$, randomly choose $x \in \mathbb{Z}_n^*$ and compute the ciphertext $c = g^M x^n \bmod n^2$. To decrypt c, compute

$$M = \frac{L(c^{\lambda(n)} \bmod n^2)}{L(g^{\lambda(n)} \bmod n^2)} \bmod n,$$

where the L function takes elements from the set $\mathcal{U}_n = \{u < n^2 \mid u = 1 \bmod n\}$ and computes $L(u) = (u-1)/n$. The semantic security is based on the difficulty to distinguish n^{th} residues modulo n^2. We refer to [35] for details.

IND-CPA Threshold Version of Paillier Cryptosystem. We recall that $\Delta = \ell!$ where ℓ is the number of servers.

Key Generation Algorithm. Choose an integer n, product of two safe primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ and $\gcd(n, \varphi(n)) = 1$. One can note that the safe prime requirement can be avoided [20] using Shoup protocol [47] without using safe primes. This allows to fully share Paillier cryptosystem from the key generation protocol to the decryption process as it appears difficult to generate RSA moduli with safe prime modulus using [10]. However, for the clarity of the description we use RSA moduli with safe primes. Set $m = p'q'$. Let β be an element randomly chosen in $\mathbb{Z}_n^*$.

The secret key $\mathsf{sk} = \beta \times m$ is shared with the Shamir scheme [46] modulo mn. Let v be a square that generates with overwhelming probability the cyclic group of squares in $\mathbb{Z}_{n^2}^*$. The verification keys vk_i are obtained with the formula $v^{\Delta \mathsf{sk}_i} \bmod n^2$.

Encryption Algorithm. To encrypt a message M, randomly pick $x \in \mathbb{Z}_n^*$ and compute $c = g^M x^n \bmod n^2$.

Partial Decryption Algorithm. The i^{th} player P_i computes the decryption share $c_i = c^{2\Delta \mathsf{sk}_i} \bmod n^2$ using his secret share sk_i. He makes a proof of correct decryption which assures that $c^{4\Delta} \bmod n^2$ and $v^{\Delta} \bmod n^2$ have been raised to the same power sk_i in order to obtain c_i^2 and vk_i.

Recovery Algorithm. If less than $t+1$ decryption shares have valid proofs of correctness the algorithm fails. Otherwise, let S be a set of $t+1$ valid shares and compute the plaintext using the Lagrange interpolation on the exponents (which is possible since exponents are multiplied by $\Delta = \ell!$, and thus no modular root extraction is required.)

In [19], they proved the following theorem.

Theorem 3. *Under the decisional composite residuosity assumption and in the random oracle model, the threshold version of Paillier cryptosystem is* IND-TCPA *against active but non-adaptive adversaries.*

Even if their definition of threshold security (the partial decryption oracles behavior) is not the same, the security result still holds within our model.

IND-CCA Threshold Version of Paillier Cryptosystem. We can therefore apply previous twin conversion.

Key Generation Algorithm. Choose, for $j = 0, 1$, an integer n_j, product of two safe primes p_j and q_j. Set $m_j = (p_j - 1)(q_j - 1)/4$. Let β_j be an element randomly chosen in $\mathbb{Z}_{n_j}^*$.

The secret keys $\mathsf{sk}_j = \beta_j \times m_j$ are shared with the Shamir scheme [46] modulo $m_j n_j$. Let v_j be a square that generates all the cyclic group of squares in $\mathbb{Z}_{n_j^2}^*$. The verification keys $\mathsf{vk}_{i,j}$ are obtained with the formula $v_j^{\Delta \mathsf{sk}_{i,j}} \bmod n_j^2$.

Encryption Algorithm. To encrypt a message M, randomly pick $x_j \in \mathbb{Z}_{n_j}^*$ and compute $a_j = g_j^M x_j^{n_j} \bmod n_j^2$. Furthermore compute a proof that a_0 and a_1 encrypt the same value: Let r be a randomly chosen element in $[0, A[$, and random elements $\alpha_j \in \mathbb{Z}_{n_j}^*$. Compute $y_j = g_j^r \alpha_j^{n_j} \bmod n_j^2$. Let e be the hash value $H(g_0, g_1, a_0, a_1, y_0, y_1)$ where H is a hash function which outputs values in the range $[0, B[$. Then, compute $z = r + e \times M$, $u_j = \alpha_j x_j^e \bmod n_j$ A proof of equality is the tuple

$$(e, z, u_0, u_1) \in [0, B[\times [0, A[\times \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^*$$

It is checked by the equation

$$e = H(g_0, g_1, a_0, a_1, g_0^z u_0^{n_0} / a_0^e \bmod n_0^2, g_1^z u_1^{n_1} / a_1^e \bmod n_1^2)$$

The decryption process is the same as in [19]. Furthermore, the above proof can be shown to be simulation-sound, using the same technique as for the El Gamal scheme, thanks to the forking lemma [39].

It is amazing to note that the Generic Conversion of Paillier cryptosystem keeps the homomorphic properties, namely that $\mathcal{E}(M_1 + M_2) \equiv \mathcal{E}(M_1) \times \mathcal{E}(M_2)$ and $\mathcal{E}(M)^k \equiv \mathcal{E}(kM)$. For example, in voting scheme, such as [15, 1], the authority can check the universally checkable proofs of validity of ciphertext and compute the tally. However, the result will no longer be a ciphertext that withstands CCA.

5 Conclusion

In this paper we have constructed generic conversions to threshold cryptosystems secure against chosen-ciphertext attacks from any cryptosystems secure against CPA. We have proposed the first version of threshold cryptosystems CCA-secure which rely on the factorization problem. A new version of Paillier cryptosystem based on a new assumption related to RSA appears in [13]. By applying our techniques, one can also share this cryptosystem under their new assumption. This provides the second threshold cryptosystem secure under CCA based on RSA.

However, as it is noted in [48], it appears to be difficult to share RSA. It seems even difficult to share OAEP-RSA without redundancy, which is a cryptosystem which achieves IND-CPA, but in the random oracle model. Indeed, the proof of membership appears to be odd and not practical.

Acknowledgement. We would like to thank Masayuki Abe for fruitful discussions.

References

1. O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical Multi-Candidate Election System. In *PODC '01*. ACM, 2001.
2. O. Baudron, D. Pointcheval, and J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In *Proc. of the 27th ICALP*, LNCS 1853, pages 499–511. Springer-Verlag, Berlin, 2000.
3. M. Bellare, A. Boldyreva, and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In *Eurocrypt '2000*, LNCS 1807, pages 259–274. Springer-Verlag, Berlin, 2000.
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, M. Fischlin, S. Goldwasser, and S. Micali. Identification Protocols Secure against Reset Attacks. In *Eurocrypt '2001*, LNCS 2045, pages 495–511. Springer-Verlag, Berlin, 2001.
6. 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.
7. 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.
8. M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *Proc. of the 20th STOC*, pages 103–112. ACM Press, New York, 1988.
9. M. Blum, P. Feldman, and S. Micali. Proving Security against Chosen-Ciphertext Attacks. In *Crypto '88*, LNCS 403, pages 256–268. Springer-Verlag, Berlin, 1989.
10. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In *Crypto '97*, LNCS 1294, pages 425–439. Springer-Verlag, Berlin, 1997.

11. R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. On Adaptive vs. Non-adaptive Security of Multiparty Protocols. In *Eurocrypt '2001*, LNCS 2045, pages 262–279. Springer-Verlag, Berlin, 2001.
12. R. Canetti and S. Goldwasser. An Efficient Threshold PKC Secure Against Adaptive CCA. In *Eurocrypt '99*, LNCS 1592, pages 90–106. Springer-Verlag, Berlin, 1999.
13. D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Q. Nguyen. Paillier's Cryptosystem Revisited. In *ACM CCS '2001*, ACM Press, 2001.
14. 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.
15. I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *PKC '2001*, LNCS 1992, pages 119–137. Springer-Verlag, Berlin, 2001.
16. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
17. 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.
18. 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.
19. P. A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Cryptography '2000*, LNCS. Springer-Verlag, Berlin, 2000.
20. P. A. Fouque and J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. In *Asiacrypt '2001*, LNCS, Springer-Verlag, Berlin, 2001.
21. P. A. Fouque and J. Stern. One Round Threshold Discrete-Log Key Generation without Private Channels. In *PKC '2001*, LNCS 1992, pages 300–316. Springer-Verlag, Berlin, 2001.
22. Y. Frankel, P. Gemmel, Ph. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In *Proc. of the 38th FOCS*, pages 384–393. IEEE, New York, 1997.
23. Y. Frankel, P. Gemmell, and M. Yung. Witness Based Cryptographic Program Checking and Robust Function Sharing. In *Proc. of the 28th STOC*, pages 499–508. ACM Press, New York, 1996.
24. Y. Frankel, P. MacKenzie, and M. Yung. Robust Efficient Distributed RSA Key Generation. In *Proc. of the 30th STOC*, pages 663–672. ACM Press, New York, 1998.
25. 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.
26. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.
27. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '2001*, LNCS. Springer-Verlag, Berlin, 2001.
28. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and Efficient Sharing of RSA Functions. In *Crypto '96*, LNCS 1109, pages 157–172. Springer-Verlag, Berlin, 1996.
29. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *Eurocrypt '96*, LNCS 1070, pages 425–438. Springer-Verlag, Berlin, 1996.
30. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Eurocrypt '99*, LNCS 1592, pages 295–310. Springer-Verlag, Berlin, 1999.
31. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
32. C.H. Lim and P.J. Lee. Another Method for Attaining Security Against Adaptively Chosen Ciphertext Attacks. In *Crypto '93*, LNCS 773, pages 287–296. Springer-Verlag, Berlin, 1994.
33. 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.
34. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '2001*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.
35. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999.
36. P. Paillier and D. Pointcheval. Efficient Public-Key Cryptosystems Provably Secure against Active Adversaries. In *Asiacrypt '99*, LNCS 1716, pages 165–179. Springer-Verlag, Berlin, 1999.
37. T. Pedersen. A Threshold Cryptosystem without a Trusted Party. In *Eurocrypt '91*, LNCS 547, pages 522–526. Springer-Verlag, Berlin, 1992.
38. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '2000*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.

39. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
40. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In *Crypto '98*, LNCS 1462, pages 89–104. Springer-Verlag, Berlin, 1998.
41. 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.
42. A. Sahai. Non-Malleable Non-Interactive Zero-Knowledge and Chosen-Ciphertext Security. In *FOCS '99*, LNCS 2139. IEEE, 1999.
43. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. In *Proc. of the 26th STOC*, pages 522–523. ACM Press, New York, 1994.
44. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
45. C. P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In *Asiacrypt '2000*, LNCS 1976, pages 458–469. Springer-Verlag, Berlin, 2000.
46. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, November 1979.
47. V. Shoup. Practical Threshold Signatures. In *Eurocrypt '2000*, LNCS 1807, pages 207–220. Springer-Verlag, Berlin, 2000.
48. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, Berlin, 1998.
49. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC '98*, LNCS. Springer-Verlag, Berlin, 1998.