This paper appears in *Financial Cryptography 2000 Proceedings, LNCS, Springer Verlag, 2000*

Sharing Decryption in the Context of Voting or Lotteries

Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern

École Normale Supérieure
Laboratoire d'informatique
45, rue d'Ulm
F-75230 Paris Cedex 05
{Pierre-Alain.Fouque, Guillaume.Poupard, Jacques.Stern}@ens.fr

Abstract. Several public key cryptosystems with additional homomorphic properties have been proposed so far. They allow to perform computation with encrypted data without the knowledge of any secret information. In many applications, the ability to perform decryption, i.e. the knowledge of the secret key, gives a huge power. A classical way to reduce the trust in such a secret owner, and consequently to increase the security, is to share the secret between many entities in such a way that cooperation between them is necessary to decrypt. In this paper, we propose a distributed version of the Paillier cryptosystem presented at Eurocrypt '99. This shared scheme can for example be used in an electronic voting scheme or in a lottery where a random number related to the winning ticket has to be jointly chosen by all participants.

1 Introduction

Public Key encryption is a central primitive in cryptology. It enables to encrypt messages using only public keys in such a way that only the owner of the corresponding secret key can perform decryption. The most famous scheme, RSA [17], is widely used but many other cryptosystems have been developed to provide additional properties. Of interest to us is a family of schemes based on a very simple encryption mechanism that essentially performs an exponentiation of the message to encrypt. The security relies on the intractability of computing discrete logarithms while knowledge of a trapdoor, the secret key, allows to efficiently decrypt ciphertexts. We call such schemes trapdoor discrete logarithm schemes. Those protocols have an interesting "homomorphic" property : the encryption of the sum of two messages is equal to the product of the encryption of each one. This can be used in applications that require computing with encrypted numbers : voting schemes, lottery protocols, etc ...

In such applications, the ability to perform decryption, i.e. the knowledge of the secret key, gives a huge power. A classical way to reduce the trust in such a secret owner, and consequently to increase the security as well as the availability, is to share the secret between many entities in such a way that cooperation between them is necessary to decrypt. A threshold decryption scheme is a protocol that allows any subset of $t+1$ out of ℓ entities, or servers, to decrypt a ciphertext, but disallows the decryption if less than t servers participate in the protocol. Threshold schemes may be used when some servers are corrupted and do not play according to their nominal behavior. For instance, an adversary can make them stop, or play with different secrets. If the decryption is still correct in the presence of an attacker who plays maliciously for the corrupted servers, we say that the protocol is robust.

1.1 Our results

In this paper, we transform an homomorphic cryptosystems into a threshold version where the decryption algorithm is shared between several servers. In order to decrypt a ciphertext, each server first computes a decryption share and then a public combining algorithm outputs the plaintext. Most homomorphic cryptosystems as Goldwasser-Micali's [11], Benaloh's [1, 4], Naccache-Stern's [12], Okamoto-Uchiyama's [13] or Paillier's [14] cryptosystems need to distribute a secret value related to the factorization of an RSA modulus. We use the recent threshold techniques developed by Shoup in [21] which allows to distribute RSA signature and we extend them to the current context.

To build on firm ground, we have to develop a new security model in order to analyze the semantic security of these schemes, which are secure against Chosen Plaintext Attack (CPA) in the context of threshold CPA-security. Previous definitions of threshold cryptosystems secure against Chosen Ciphertext Attack (CCA) have been formalized as a natural extension of the standard definitions of CCA-security in [9]. Following this work, we propose adequate definitions to assert the CPA-security of threshold cryptosystems.

Homomorphic cryptosystems have been used in electronic voting schemes with multiple authorities among which the decryption process is distributed [5]. The previously proposed schemes use a variant of the ElGamal encryption scheme : instead of encrypting m with (g^k, my^k), it computes $(g^k, g^m y^k)$. Unfortunately, such a scheme cannot be considered as a trapdoor discrete logarithm scheme because no trapdoor exists to determine m given $g^m \bmod p$. Anyway, in voting schemes, the cryptosystems only manage small numbers because the number of voters is restricted and each voter votes "0" or "1". Consequently, the tally cannot be very large and an exhaustive search allows to give the result.

However, in some circumstances, it is useful to decrypt larger numbers. In such applications, the modified ElGamal scheme can no longer be used since exhaustive search, or more efficient methods like Pollard's rho algorithm, cannot recover the plaintext. A solution is to use a threshold trapdoor discrete logarithm scheme. Below, we describe two applications where the decryption of large numbers is necessary, but other applications may be found.

1.2 Applications

We present applications where our threshold decryption scheme can be used to efficiently recover "large" plaintext for homomorphic encryption schemes. We show how one can use this primitive to build a multiple voting scheme or a lottery scheme.

Multiple election schemes. Threshold trapdoor discrete logarithm cryptosystems can be used to distribute an electronic election between multiple authorities, as proposed in [5]. With our threshold trapdoor discrete logarithm scheme, we can determine the tally directly from the trapdoor. Systems that require exhaustive search at a point of the decryption phase are not able to solve some situations. For example, if we want to make multiple election, we can use the following mechanism and our decryption technique:

Let N be the number of voters and k such that $N < 2^k$. The voters vote "1" for the first candidate, "2^k" for the second, "2^{2k}" for the third, and so on. It is easy to show that the first k bits give the result of the first candidate, the following k bits give the result of the second candidate, etc. In this case, if the number of candidate is high, exhaustive search can no longer be used. With our threshold decryption scheme, we are able to manage $\lfloor |n|/k \rfloor$ candidates, where $|n|$ is the size of the RSA modulus.

A lottery scheme. A publicly verifiable decryption of a large number can also be useful in a lottery scheme. Consider a lottery which have to compute a random number in order to indicate the winning ticket in the range $\{0, \ldots N-1\}$, where N is the number of players. In a typical lottery, one or more winners are chosen during a trusted process so that each purchased ticket has an equal chance to be chosen. This process is usually monitored by an outsider auditor which ensures the fairness of the protocol. As the process is random, it cannot be repeated and ticket purchasers must trust the process.

Previous schemes, developed by Goldschlag and Stubblebine [10], use *Delaying Functions* to prevent computation of the result by the lottery or anybody else before the end of the purchase phase. In these protocols, all players have access to the random inputs of the delaying function as assumptions made on such functions ensures that nobody can compute the output before a certain time in the future.

We use the same framework as [10] but we do not use delaying function. The security of our lottery scheme is only based on standard assumptions. As previous schemes, our lottery uses random numbers chosen by the players in order to output a number whose randomness is granted provided at least one player chooses its number at random. Each random number is encrypted by the players with the homomorphic cryptosystem of the lottery. Thus, nobody, except the lottery, can learn the random inputs. Moreover, we share the decryption process between servers so that the lottery itself cannot compute the final result even if t out of ℓ servers play maliciously and try to recover the result before the end of the purchase phase.

The homomorphic property allows to compute efficiently the encryption of the sum s of the random numbers modulo n. Next, a quorum of at least $t+1$

servers must collaborate to obtain $s \bmod n$. Finally, $(s \bmod n) \bmod N$ corresponds to a sequence number generated by the lottery when a player sends its random value. This is the winning ticket of the lottery. Furthermore, as we use a threshold decryption process which is publicly verifiable, the servers' operations can be verified by the players.

1.3 Notations and Definitions

Throughout this paper, we use the following notation: for any integer n,

- we use $\mathbb{Z}_n$ to denote the set of the integers modulo n,
- we use ${\mathbb{Z}_n}^*$ to denote the multiplicative group of invertible elements of $\mathbb{Z}_n$,
- we use $\varphi(n)$ to denote the Euler totient function, i.e. the cardinality of ${\mathbb{Z}_n}^*$,
- we use $\lambda(n)$ to denote Carmichael's lambda function defined as the largest order of the elements of ${\mathbb{Z}_n}^*$.

It is well known that if the prime factorization of an odd integer n is $\prod_{i=1}^{\eta} {q_i}^{f_i}$ then $\varphi(n) = \prod_{i=1}^{\eta} {q_i}^{f_i-1}(q_i - 1)$ and $\lambda(n) = \mathrm{lcm}_{i=1\ldots\eta}\left({q_i}^{f_i-1}(q_i - 1)\right)$.

Finally, a prime number p is said to be a *strong prime* if $p = 2p' + 1$ and p' is also prime.

1.4 Outline of the paper

In section 2 we give a formal definition of a threshold cryptosystem. We also propose definition of semantic security adapted to those schemes. Then, in sections 3 and 4, we describe useful tools and Shoup's threshold RSA signature scheme. Finally, in section 5, we distribute the homomorphic encryption scheme of Paillier and we prove its security under the same assumption as the original version.

2 Definition and security of threshold cryptosystems

2.1 Formal definition

A threshold cryptosystem consists of the four following components :

- A *key generation algorithm* takes as input a security parameter k, the number ℓ of decryption servers, the threshold parameter t and a random string ω; it outputs a public key PK, a list $SK_1, \ldots SK_\ell$ of private keys and a list $VK, VK_1, \ldots VK_\ell$ of verification keys.
- An *encryption algorithm* takes as input the public key PK, a random string ω and a cleartext M; it outputs a ciphertext c.
- A *share decryption algorithm* takes as input the public key PK, an index $1 \leq i \leq \ell$, the private key SK_i and a ciphertext c; it outputs a decryption share c_i and a proof of its validity $proof_i$.

– A *combining algorithm* takes as input the public key PK, a ciphertext c, a list $c_1, \dots c_\ell$ of decryption shares, the list $VK, VK_1, \dots VK_\ell$ of verification keys and a list $proof_1, \dots proof_\ell$ of validity proofs; it outputs a cleartext M or fails.

2.2 The players and the scenario

Our game includes the following players : a dealer, a combiner, a set of ℓ servers P_i, an adversary and users. All are considered as probabilistic polynomial time Turing machines. We consider the following scenario :

– In an initialization phase, the dealer uses the key generation algorithm to create the public, private and verification keys. The public key PK and all the verification keys VK, VK_i are publicized and each server receives its share SK_i of the secret key SK.
– To encrypt a message, any user can run the encryption algorithm using the public key PK.
– To decrypt a ciphertext c, the combiner first forwards c to the servers. Using their secret keys SK_i and their verification keys VK, VK_i, each server runs the decryption algorithm and outputs a partial decryption c_i with a proof of validity of the partial decryption $proof_i$. Finally, the combiner uses the combining algorithm to recover the cleartext if enough partial decryptions are valid.

2.3 Security requirements

We consider an adversary able to corrupt up to t servers. Such a corruption can be passive, i.e. the attacker only eavesdrops the servers. It can also consist in making the servers fail and stop. Finally, it can be active; in this last case, the adversary completely controls the behavior of the corrupted servers. In the following, we only consider non-adaptive adversaries who choose which servers they want to corrupt before key generation.

A threshold cryptosystem is said to be *t-robust* if the combiner is able to correctly decrypt any ciphertext, even in the presence of an adversary who actively corrupts up to t servers.

All messages are sent in clear between each server and the combiner. Moreover, the combining algorithm which takes each partial decryption and recovers the cleartext is public and can be executed by any server as they see all decryption parts. So the only assumption we make about the communication channel is the existence of a broadcast channel between all participants.

Threshold semantic security. All the encryption schemes we study are semantically secure. Informally speaking, let us consider an attacker who first issues two messages M_0 and M_1; we randomly choose one of these messages, we encrypt it and we send this ciphertext to the attacker. Finally, she answers which message has been encrypted. We say that the encryption scheme is semantically secure if there exists no such polynomial time attacker able to guess which of the two messages has been encrypted with a non-negligible advantage.

We extend the definition of semantic security to threshold cryptosystems in the setting where an attacker who actively, but non-adaptively, corrupts t servers learns the public parameters, as in the regular cryptosystem but also the secret keys of the corrupted servers, the public verification keys, all the decryption shares and the proof of validity of those shares.

Let us consider the following game A:

A1 The attacker chooses to corrupt t servers. She learns all their secret information and she actively controls their behavior.
A2 The key generation algorithm is run; the public keys are publicized, each server receives its secret keys and the attacker learns the secrets of the corrupted players.
A3 The attacker chooses a message M and a *partial decryption oracle* gives her ℓ valid decryption shares of the encryption of M, along with proofs of validity. This step is repeated as many times as the attacker wishes.
A4 The attacker issues two messages M_0 and M_1 and sends them to an *encryption oracle* who randomly chooses a bit b and sends back an encryption c of M_b to the attacker.
A5 The attacker repeats step A3, asking for decryption shares of encryptions of chosen messages.
A6 The attacker outputs a bit b'.

A threshold encryption scheme is said to be semantically secure against active non-adaptive adversaries if for any polynomial time attacker, $b = b'$ with probability only negligibly greater than $1/2$.

Notice that our definition of semantic security reduces to the original one when we consider only one server ($\ell = 1$) who knows the secret key, and an adversary who does not corrupt any server ($t = 0$). In this case, steps A3 and A5 just consists into encrypting chosen plaintexts and this can be done without the help of a *partial decryption oracle*.

Finally, the previous game may not be confused with the chosen ciphertext attack security described by Gennaro and Shoup [9]. The attacker can only ask for partial decryptions of ciphertexts for which she already knows the corresponding plaintext. The goal of steps A3 and A5 is to prove that partial decryptions give no information about the private keys of the non-corrupted servers. Since the cryptosystems we study are not immunized against chosen ciphertext attacks in the non-distributed case, we cannot expect to extend such a property to threshold versions.

Security Proofs. Our aim is to provide robust threshold version of semantically secure cryptosystems. Our security proofs are based on reduction ; we prove that if an adversary can break the semantic security of the threshold cryptosystem, then she must be able to break the semantic security of the initial cryptosystem.

We show how to build an adversary to attack the semantic security of the traditional cryptosystem from an adversary who can break the security of the threshold cryptosystem. The basic idea in order to use an attacker against the threshold version and to turn her into an attacker against the traditional cryptosystem, is to simulate all the extra information that are not provided in a traditional attack.

More precisely, if we are able to simulate the public verification keys and of the secret keys of corrupted servers in step A2, the decryption shares and their proof of validity in steps A3 and A5 in such a way that the adversary cannot distinguish between the real distribution and the simulated one, we can feed this adversary with simulated data in order to obtain an attacker against the semantic security of the initial non-distributed scheme. Consequently, the security proofs consist into showing how to simulate all those data.

3 Preliminary tools

3.1 Shamir threshold secret sharing scheme

In 1979, Shamir [20] proposed a protocol to share a secret element s of a field F between ℓ servers in such a way that any group of $t+1$ servers can efficiently recover s but any coalition of t servers are not able to gain any information about the secret.

The scheme is based on the Lagrange interpolation formula that allows to compute $P(X)$, for any $X \in F$, if P is a polynomial of degree t and if $t+1$ values $P(x_i)$ are known for $t+1$ distinct elements $x_1, \ldots x_{t+1}$ of F:

$$P(X) = \sum_{i=1}^{t+1} \prod_{\substack{j=1 \\ j \neq i}}^{t+1} \frac{X - x_j}{x_i - x_j} \times P(x_i)$$

In order to share a secret s, a randomly chosen polynomial P of degree t is chosen in $F[X]$ such that $P(0) = s$ and each server receives a point $(x_i, P(x_i))$ with $x_i \neq 0$. The Lagrange formula shows that the knowledge of $t+1$ such points allows to recover the all polynomial P and consequently $P(0) = s$. One can also prove that knowledge of t points gives no information about s.

3.2 Proof of equality of discrete logarithms in a cyclic group of unknown order

Let $\mathcal{G}$ be a cyclic group of unknown order m. Let g be a generator of $\mathcal{G}$ and h and element of $\mathcal{G}$. A proof of equality of the discrete logarithm of an element G in the basis g and of another element H in the basis h can be designed using the Chaum and Pedersen protocol [3] in such a way that the order of $\mathcal{G}$ does not have to be known. This is possible using a variant of the Schnorr proof of knowledge of discrete log [19] proposed in [15, 2].

We now describe the non-interactive version of the proof. Let s be this common discrete logarithm. let r be a randomly chosen element in $[0, A[$. Compute

$x = g^r$ and $x' = h^r$. Let e be the hash value $H'(g, h, G, H, x, x')$ where H' is an hash function which outputs values in the range $[0, B[$. Then, compute $y = r + e \times s$. A proof of equality of discrete logs is such a pair $(e, y) \in [0, B[\times[0, A[$; it is checked by the equation $e = H'(g, h, G, H, g^y/G^e, h^y/H^e)$.

The correctness of such a scheme is obvious. Furthermore, we can prove that if A is much larger than $B \times m$, the protocol statistically gives no information about the secret. Finally, let us focus on soundness; we remind the security proof of [21] in the random oracle model. If a proof (e, y) is valid, we have $e = H'(g, h, G, H, g^y/G^e, h^y/H^e)$. Let $x = g^y/G^e$ and $x' = h^y/H^e$. Since $\mathcal{G}$ is a cyclic group generated by g, let a, b, c and d be integers such that $h = g^a$, $H = g^b$, $x = g^c$ and $x' = g^d$. Using the definition of x and x', we obtain the equations $c = y - se \bmod m$ and $d = ay - be \bmod m$ so, multiplying the first one by a and subtracting the second one, $ca - d = e(b - sa) \bmod m$. In the random oracle model, e is a random value, independent of the inputs of the hash function so necessarily $b - sa = 0 \bmod m$ and, finally, $g^b = H = h^s = g^{as}$. Consequently G and H have the same discrete logarithm in the respective basis g and h.

3.3 The Paillier cryptosystem

Various cryptosystems based on randomized encryption schemes $E(M)$ which encrypt a message M by raising a basis g to the power M have been proposed so far [11, 1, 4, 22, 12–14]. Their security is based on the intractability of computing discrete logarithm in the basis g without a secret data, the secret key, and easy using this trapdoor. We call those cryptosystems *trapdoor discrete logarithm schemes*. As an important consequence of this encryption technique, those schemes have homomorphic properties that can be informally stated as follows:

$$E(M_1 + M_2) = E(M_1) \times E(M_2) \quad \text{and} \quad E(k \times M) = E(M)^k$$

Paillier has presented three closely related such cryptosystems in [14]. We only remind the first one. This 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 $w \in \mathbb{Z}_{n^2}{}^*$,

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

Key Generation. 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 $PK = (n, g)$ and the secret key is $SK = \lambda(n)$.

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

Decryption. To decrypt c, compute $M = \dfrac{L(c^{\lambda(n)} \bmod n^2)}{L(g^{\lambda(n)} \bmod n^2)} \bmod n$ where the L-function takes in input elements from the set $\mathcal{S}_n = \{u < n^2 | u = 1 \bmod n\}$ and computes $L(u) = \frac{u-1}{n}$.

The integers $c^{\lambda(n)} \bmod n^2$ and $g^{\lambda(n)} \bmod n^2$ are equal to 1 when they are raised to the power n so they are n^{th} roots of unity. Furthermore, such roots are of the form $(1+n)^\beta = 1 + \beta n \bmod n^2$. Consequently, the L-function allows to compute such values $\beta \bmod n$ and $L((g^M)^{\lambda(n)} \bmod n^2) = M \times L(g^{\lambda(n)} \bmod n^2) \bmod n$.

Security. It is conjectured that the so-called composite residuosity class problem, that exactly consists in inverting the cryptosystem, is intractable. The semantic security is based on the difficulty to distinguish n^{th} residues modulo n^2. We refer to [14] for details.

4 Threshold version of RSA cryptosystem

In this section, we recall previous works about threshold RSA and describe Shoup's threshold RSA signature scheme [21] that can also be viewed as a threshold RSA decryption algorithm. This scheme is indeed a starting point for the design of efficient threshold trapdoor discrete log cryptosystem proposed in the next section.

Desmedt and Frankel have been pioneers in threshold cryptography and proposed in [6] a threshold RSA signature protocol using Shamir's polynomial secret sharing scheme in the ring $\mathbb{Z}_{\lambda(n)}$. In order to hide the inverses, Desmedt and Frankel [6], followed by de Santis *et al* [18] and Gennaro *et al* [8] extend the ring of integers modulo $\varphi(n)$ to another algebraic structure, a module, where the inverses can be disclosed safely. Other tentatives of Frankel *et al* [7], followed by Rabin [16], have been made to avoid the use of strong primes as factors of n.

Recently, Shoup [21] has solved this problem with a much simpler and elegant scheme. Let us remind the protocol :

Key generation algorithm. The dealer chooses two strong primes $p = 2p' + 1$ and $q = 2q' + 1$; the RSA modulus is $n = pq$ and the public exponent e is a prime number greater than ℓ: $PK = (n, e)$. Let m be $p' \times q'$. The dealer then computes the secret key $d \in \mathbb{Z}_m$ such that $de = 1 \bmod m$. It is shared using Shamir's secret sharing scheme: let $f_0 = d$ and, for $i = 1, \ldots t$, f_i is randomly chosen in $\mathbb{Z}_m$. Let $f(X) = \sum_{i=0}^{t} f_i X^i$; the secret key SK_i is $d_i = f(i) \bmod m$. Finally, the dealer randomly chooses v in the cyclic subgroup of squares in $\mathbb{Z}_n{}^*$ and computes the verification keys $VK = v$ and, for $i = 1 \ldots \ell$, $VK_i = v^{d_i} \bmod n$.

Encryption algorithm. In order to encrypt a message M, any user can compute $c = M^e \bmod n$.

Share decryption algorithm. Let Δ be $\ell!$; in order to obtain his share of the decryption of a ciphertext c, each server computes $c_i = c^{2\Delta d_i} \bmod n$ and a proof of validity to convince the combiner, or anybody else, that the discrete logarithm of c_i^2 in the base $c^{4\Delta}$ is the same as the discrete log of VK_i in the base VK, namely the secret key $SK_i = d_i$. Such a non-interactive proof is proposed in section 3.2.

Combining algorithm. If less than t decryption shares have valid proofs of correctness the algorithm fails. Otherwise, let S be a set of $t + 1$ valid shares;

for any $i \in \{0, \ldots \ell\}$ and $j \in S$, let us define a variant of Lagrange coefficients :

$$\mu_{i,j}^S = \Delta \times \frac{\prod_{j' \in S \setminus \{j\}} (i - j')}{\prod_{j' \in S \setminus \{j\}} (j - j')} \in \mathbb{Z}$$

The Δ factor is used in order to obtain integers and to avoid the computation of inverses modulo the secret value m. Therefore, the Lagrange interpolation formula implies :

$$\Delta f(i) = \sum_{j \in S} \mu_{i,j}^S f(j) \bmod m$$

Using the fact that $f(0)$ is equal to the secret key d, we can partially decrypt c :

$$M^{4\Delta^2} = c^{4\Delta^2 d} = c^{4\Delta^2 f(0)} = \prod_{j \in S} c^{4\Delta \mu_{0,j}^S f(j)} = \prod_{j \in S} c_i^{2\mu_{0,j}^S} \bmod n$$

Finally, since the public exponent is a prime number greater than ℓ, it is relatively prime with $4\Delta^2$ so the extended Euclidean algorithm provides integers a and b such that $a \times 4\Delta^2 + b \times e = 1$. This allows to obtain the plaintext $M = M^{a \times 4\Delta^2} \times M^{b \times e} = \left(M^{4\Delta^2}\right)^a \times c^b \bmod n$.

5 Threshold version of Paillier cryptosystem

5.1 Description

We recall that $\Delta = \ell!$ where ℓ is the number of servers.

Key generation algorithm. Choose an integer n, product of two strong primes p and q, such that $p = 2p' + 1$ and $q = 2q' + 1$ and $\gcd(n, \varphi(n)) = 1$. Set $m = p'q'$. Let β be an element randomly chosen in ${\mathbb{Z}_n}^*$. Then randomly choose $(a, b) \in {\mathbb{Z}_n}^* \times {\mathbb{Z}_n}^*$ and set $g = (1+n)^a \times b^n \bmod n^2$. The secret key $SK = \beta \times m$ is shared with the Shamir scheme: let $a_0 = \beta m$, randomly choose t values a_i in $\{0, \ldots n \times m - 1\}$ and set $f(X) = \sum_{i=0}^t a_i X^i$. The share s_i of the i^{th} server P_i is $f(i) \bmod nm$. The public key PK consists of g, n and the value $\theta = L(g^{m\beta}) = am\beta \bmod n$. Let $VK = v$ be a square that generates of the cyclic group of squares in ${\mathbb{Z}_{n^2}}^*$. The verification keys VK_i are obtained with the formula $v^{\Delta s_i} \bmod n^2$.

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

Share decryption algorithm. The i^{th} player P_i computes the decryption share $c_i = c^{2\Delta s_i} \bmod n^2$ using his secret share s_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 s_i in order to obtain c_i^2 and v_i (see protocol of section 3.2).

Combining algorithm. If less than t 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

$$M = L\left(\prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod n^2\right) \times \frac{1}{4\Delta^2\theta} \bmod n$$

where $\mu_{0,j}^S = \Delta \times \prod_{j'\in S\setminus\{j\}} \frac{j'}{j'-j} \in \mathbb{Z}$

Notes on the correctness of the scheme. First notice that we choose n such that $\gcd(n, \varphi(n)) = 1$. This condition ensures that the function defined by $f(a,b) = (1+n)^a \times b^n \bmod n^2$ is a bijection from $\mathbb{Z}_n \times {\mathbb{Z}_n}^*$ to ${\mathbb{Z}_{n^2}}^*$.

The order of g in ${\mathbb{Z}_{n^2}}^*$ is $n \times \alpha$ where α is the order of b in ${\mathbb{Z}_n}^*$. Furthermore, we can see that the subgroup of the squares in ${\mathbb{Z}_{n^2}}^*$ is cyclic and that its order is nm. The number of generators of this group is $\varphi(nm)$ so the probability for a randomly chosen square in ${\mathbb{Z}_{n^2}}^*$ to be a generator is about $1 - 1/\sqrt{n}$ and this probability is overwhelming. Then, the verification keys VK_i may be seen as witnesses of the knowledge of a discrete log of VK_i in base $v^\Delta \bmod n^2$. They are used to make for proofs of validity for partial decryptions.

Finally, let us consider a subset S of $t+1$ correct shares; the computation of $c^{4\Delta^2 m\beta}$ can be done using the Lagrange interpolation formula:

$$\Delta f(0) = \Delta m\beta = \sum_{j\in S} \mu_{0,j}^S f(j) \bmod nm$$

so

$$c^{4\Delta^2 m\beta} = \prod_{j\in S} c^{4\Delta s_j \mu_{0,j}^S} = \prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod n^2$$

If c is an encryption of a message M,

$$c^{4\Delta^2 m\beta} = g^{4\Delta^2 M\beta m} = (1+n)^{4\Delta^2 Mam\beta} = 1 + 4\Delta^2 Mam\beta n \bmod n^2$$

Consequently, $L\left(\prod_{j\in S} c_j^{2\mu_{0,j}^S} \bmod n^2\right) = 4M\Delta^2 am\beta = M \times 4\Delta^2\theta \bmod n$. As θ is part of the public key, we obtain the message M.

5.2 Proof of Security

In the following, we only use modulus n such that $\gcd(n, \varphi(n)) = 1$. Let us denote $CR[n]$ the problem of deciding n^{th} residuosity, i.e. distinguishing n^{th} residues from non-n^{th} residues. The semantic security of Paillier scheme with modulus n is equivalent to $CR[n]$ (see [14] for more details). In the following, we refer to the so-called *Decisional Composite Residuosity Assumption* (DCRA) which assumes that $CR[n]$ is intractable.

We now prove that the threshold version of the Paillier scheme is secure according to the definition of threshold semantic security proposed in section 2.3. Basically, such a proof shows that if an attacker is able to break the threshold semantic security, it can be used to break the semantic security of the original cryptosystem.

More precisely, as we said in section 2.3, we have to simulate information that an attacker may obtained in steps A2, A3, A4, A5 and A6. In step A2, we simulate parameters of the threshold cryptosystem. In steps A3 and A5, we simulate the decryption parts of non-corrupted servers along with correctness proofs. Finally, steps A4 and A6 represent the two steps of the standard definition of semantic security for non-threshold cryptosystems.

Theorem 1. *Under the decisional composite residuosity assumption and in the random oracle model, the threshold version of Paillier cryptosystem is semantically secure against active non-adaptive adversaries.*

Proof. Let us assume the existence of an adversary $\mathcal{A}$ able to break the semantic security of the threshold scheme. We now describe an attacker which uses $\mathcal{A}$ in order to break the semantic security of the original Paillier scheme. In a first phase, called the find phase, the attacker obtains the public key (n, g) and he chooses two messages M_0 and M_1 which are sent to an encryption oracle who randomly chooses a bit b and returns an encryption c of M_b. In a second phase, called the guess phase, the attacker tries to guess which message has been encrypted.

We now describe how to feed an adversary $\mathcal{A}$ of the threshold scheme in order to make a semantic attacker. In step A1 of game A, the adversary chooses to corrupt t servers. Without loss of generality, we assume that the first t servers $P_1, \ldots P_t$ are corrupted.

In the find phase, the attacker first obtains the public key $PK = (n, g)$ of the regular Paillier scheme. He randomly chooses $(a_1, b_1, \theta) \in {\mathbb{Z}_n}^* \times {\mathbb{Z}_n}^* \times {\mathbb{Z}_n}^*$ and he sets $g_1 = g^{a_1} \times b_1^n \bmod n^2$. He also picks at random t values $s_1, \ldots s_t$ in the range $\{0, \ldots \lfloor n^2/4 \rfloor\}$, a randomly chosen element α of ${\mathbb{Z}_n}^*$ and sets $v = g_1^{2\alpha} \bmod n^2$.

Then, he computes $v_i = v^{\Delta s_i} \bmod n^2$, for $i = 1, \ldots t$, and the other verification keys as

$$v_i = (1 + 2\alpha\theta n)^{\mu_{i,0}^S} \times \prod_{j \in S \setminus \{0\}} v^{s_j \mu_{i,j}^S} \bmod n^2$$

where $S = \{0, 1, \ldots t\}$. The attacker sends $(n, g_1, \theta, v, v_1, \ldots v_\ell, s_1, \ldots s_t)$ to $\mathcal{A}$ in step A2 of game A.

During step A3, $\mathcal{A}$ chooses a message M and sends it to the attacker. He computes $c = g_1^M x^n \bmod n^2$, a valid encryption of M. The decryption shares of the corrupted players are correctly computed using the s_i's: $c_i = c^{2\Delta s_i} \bmod n^2$, for $i = 1, \ldots t$. The other shares are obtained by interpolation as

$$c_i = (1 + 2M\theta n)^{\mu_{i,0}^S} \times \prod_{j \in S \setminus \{0\}} c^{2 s_i \mu_{i,j}^S} \bmod n^2$$

Finally, the attacker chooses e at random in $[0, B[$, y at random in $[0, A[$ and sets $proof_i = (e, y)$. He returns $(c, c_1, \ldots c_\ell, proof_1, \ldots proof_\ell)$.

In step A4, $\mathcal{A}$ chooses and outputs two messages M_0 and M_1. The attacker outputs those two messages as the result of the find phase.

Then an encryption oracle for the non-threshold Paillier scheme chooses a random bit and sends an encryption c of M_b to the attacker. He computes $\Gamma = c^{a_1} \bmod n^2$ and sends Γ to the adversary $\mathcal{A}$.

Step A5 is similar to step A3. Finally, in step A6, $\mathcal{A}$ answers a bit b' which is returned by the attacker in the guess phase.

We now prove that all the data simulated by the attacker cannot be distinguished from real ones by $\mathcal{A}$. Consequently, if there exists a polynomial time adversary $\mathcal{A}$ able to break the semantic security of the threshold scheme, we have made an attacker able to break the semantic security of the original Paillier scheme.

Indistinguishability of data received by $\mathcal{A}$ during step A2.
Firstly we observe that $g_1 = g^{a_1} b_1^n \bmod n^2$ is uniformly distributed in the set of the elements of order multiple of n, provided the order of g is also a multiple of n. We need to perform such a modification of g because we choose v as an even power of g_1 and we want v to generate the subgroup of squares modulo n^2. Consequently, g has to be randomized in order to obtain, with very high probability, a basis of very large order. As an example, the valid basis $g = 1 + n$ would obviously never lead to a correct v.

We also notice that θ and v are uniformly distributed respectively in $\mathbb{Z}_n{}^*$ and Q_{n^2}, the set of the squares modulo n^2. Furthermore, v is a generator of Q_{n^2} with overwhelming probability: the statistical distance between the uniform distribution on the subset of generators of Q_{n^2}, of order $\varphi(nm)$, and the uniform distribution on Q_{n^2}, of order nm, is $O(n^{-1/2})$.

Then, the attacker chooses the secret keys $s_1, \dots s_t$ of the corrupted players; s_i should be in the interval $\{0, \dots nm\}$ but, since m is unknown, we pick s_i in $\{0, \dots \lfloor n^2/4 \rfloor\}$. Anyway, the statistical distance between the uniform distribution on $\{0, \dots \lfloor n^2/4 \rfloor - 1\}$ and the uniform distribution on $\{0, \dots nm - 1\}$ is $O(n^{-1/2})$ so the adversary cannot distinguish real and simulated corrupted secret keys.

When the dealer correctly distributes the shares, the two following conditions hold:

– For any set S of size $t+1$ and for any $i \notin S$, $v_i^{\Delta} = \prod_{j \in S} v_j^{\mu_{i,j}^S} \bmod n^2$
– For any S of size $t+1$, $\prod_{j \in S} v_j^{\mu_{0,j}^S} \bmod n^2 \in \{u < n^2 | u = 1 \bmod n\}$

In the simulation, we choose $v^{m\beta} = 1 + 2\alpha\theta n \bmod n^2$ without knowing m but just randomly choosing θ. The verification keys of corrupted servers are computed using the known secret keys s_i and the missing v_i's are obtained with the Lagrange interpolation formula. Of course, we are not able to find the missing secret keys but in fact we do not need them. So the distribution received by $\mathcal{A}$ during the key generation step is indistinguishable from a real distribution.

Indistinguishability of data received by $\mathcal{A}$ during steps A3 and A5.
In steps A3 and A5, an encryption of the message M is first computed: $c = g_1^M x^n \bmod n^2$. Then the shares of the corrupted players $c_1, \dots c_t$ are computed

using the secret keys $s_1, \dots s_t$ as $c_i = c^{2\Delta s_i} \bmod n^2$. Finally, the missing c_i's are obtained by interpolation, like the v_i's, using $c_1, \dots c_t$ and the $(t+1)^{\text{th}}$ point $c^{m\beta} \bmod n^2$ which we can compute without any secret knowledge since it is equal to $1 + 2M\theta n$.

Finally, in the proof simulation, the distribution produced by the attacker is statistically close to perfect as it is remained in section 3.2. This simulation previously appeared in [21]. In the random oracle model where the attacker has a full control of the values returned by the hash function H, we define the value of H at $(v, c^{4\Delta}, v_i, c_i^2, v^y/v_i^e, c^{4\Delta^2 y}/c_i^{2e})$ to be e. With overwhelming probability, the attacker has not yet defined the random oracle at this point so the adversary $\mathcal{A}$ cannot detect the fraud. $\square$

6 Conclusion

In this paper, we have proposed a threshold distributed version of the Paillier cryptosystem [14]. We think that this scheme is the most interesting trapdoor discrete logarithm cryptosystem, according to its efficiency and to its large bandwidth. In order to study the security of our proposal, we have defined semantic security for threshold cryptosystems.

The distribution of other trapdoor discrete logarithm cryptosystems [11, 1, 4, 22, 12, 13] still remains an open problem.

References

1. J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, 1987.
2. J. Camenisch and M. Michels. A Group Signature Scheme with Improved Efficiency. In *Asiacrypt '98*, LNCS 1514. Springer-Verlag, 1998.
3. D. Chaum and T. P. Pedersen. Wallet Databases with Observers. In *Crypto '92*, LNCS 740, pages 89–105. Springer-Verlag, 1992.
4. J. Cohen and M. Fisher. A robust and verifiable cryptographically secure election scheme. In *Symposium on Foundations of Computer Science*. IEEE, 1985.
5. R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Eurocrypt '97*, LNCS 1233, pages 113–118. Springer-Verlag, 1997.
6. Y. Desmedt and Y. Frankel. Parallel reliable threshold multisignature. Technical report, Department of E.E. and C.S. University of Wisconsin-Milwaukee, April 1992. TR-92-04-02.
7. Y. Frankel, P. Gemmel, Ph. MacKenzie, and M. Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In *Proc. 38th FOCS*, pages 384–393. IEEE, 1997.
8. 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, 1996.
9. R. Gennaro and V. Shoup. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, 1998.
10. D.M. Goldschlag and S.G. Stubblebine. Publicly Verifiable Lotterie : Applications of Delaying Functions. In *Financial Crypto '98*, LNCS 1465, pages 214–226. Springer-Verlag, 1998.

11. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28, 1984.
12. D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proc. of the 5th CCCS*. ACM press, 1998.
13. T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998.
14. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, 1999.
15. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Eurocrypt '98*, LNCS 1403, pages 422–436. Springer-Verlag, 1998.
16. T. Rabin. A Simplified Approach to Threshold and Proactive RSA. In *Crypto '98*, LNCS 1462, pages 89–104. Springer-Verlag, 1998.
17. R.L. Rivest, A. Shamir, and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystem. *Communications of the ACM*, 21(2):120–126, 1978.
18. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 522–523. ACM, 1994.
19. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, 1990.
20. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, Nov. 1979.
21. V. Shoup. Practical Threshold Signatures. Technical report, IBM, 1999. IBM Research Report RZ 3121.
22. S. Vanstone and R. Zuccherato. Elliptic Curve Cryptosystem Using Curves of Smooth Order Over the Ring Z_n. *IEEE Transaction on Information Theory*, IT-43, 1997.