Round Optimal Distributed Key Generation of Threshold Cryptosystem Based on Discrete Logarithm Problem

Rui Zhang and Hideki Imai

Information & Systems, Institute of Industrial Science, University of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505, Japan
zhang@imailab.iis.u-tokyo.ac.jp, imai@iis.u-tokyo.ac.jp

Abstract. We propose a distributed key generation protocol for discrete logarithm problem based threshold cryptosystems by introducing an efficient (publicly) verifiable encryption scheme from any homomorphic encryption with a non-interactive proof of fairness. Previous constructions of the same kind are either only based on a narrow definition of homomorphism or only a unique encryption scheme is considered. Our construction generalizes the scope of such design to a broader range of encryption schemes with efficient constructions of proofs of fairness. Since the protocol is round optimal (one-round) in the distributed fashion, adaptive adversary is not different from a static adversary, thus a simplified protocol design is possible. Our scheme is extremely capable for an environment with already built public key infrastructure. The verifiable encryption with fairness developed here can be used as building blocks of a variety of cryptographical applications like publicly verifiable secret sharing (PVSS), e-voting and auction schemes.

1 Introduction

A threshold cryptosystem distributes secret information among a set S of servers to build a fault-tolerant system, where there is a common public key pk and a number of secret key shares $\{sk_i\}_{i\in S}$ held by players in the group respectively. The players can cooperate to reconstruct the secret key, or even without it, sign messages or decrypt a ciphertext encrypted by pk, so that partial malfunctioning of total users will not affect the correctness of the output. A great proportion of solutions to multiparty protocols turns out to be a crux of threshold cryptosystem scheme in constructing a distributed TTP: key recovery [24], signature escrow [15,23], contract signing [27], fair exchange of digital signatures [2], e-voting [9,19] and auction [6] schemes.

One of the toughest tasks in constructing such fault-tolerant systems is the distributed key generation. A centralized server acting as a Trusted Party often becomes the target of attackers. A distributed key generation, i.e., each server contributes to the random secret key, is more important and desirable. Previous schemes of such threshold key generation have often adopted a multiparty-computation-fashioned protocol to distribute secret key shares among players

J. Zhou, M. Yung, Y. Han (Eds.): ACNS 2003, LNCS 2846, pp. 96–110, 2003.

against so-called adaptive adversaries. However, these schemes need intricate design techniques with inborn complexity in communication rounds: in a sharing phase the shares are sent to share holders via fully connected pair-wise secure channels and an additional complaining phase is needed to detect cheaters among a dealer and share holders via a broadcast channel. Shares' secrecy can be protected by secure channels, however, the cheater somehow survives even after the key generation, when some player accuses the dealer, a third player cannot distinguish between these two facts:

- The dealer has distributed a false share.
- The share holder lies on a good share.

In this paper, we try to solve this problem by proposing a non-interactive key generation under general computational assumption, where the share holder has no chance to misbehave.

1.1 Related Work

Threshold cryptosystems have been studied in [10,11,31,8,22], one common property of these schemes is that they work only with an honest majority. In [10, 11] Desmedt et al. define the concept of threshold schemes and propose the first implementation. However, this yields no provably secure solution. The first provable secure practical scheme is proposed by Shoup and Gennaro in [31] based on the Diffie-Hellman problem [12] in the Random Oracle model [13,4], which is resilient against a static adversary. Later research has focused on stronger attack model: in [21], Lysyanskaya et al. propose two new models of security for this kind of attacks namely the concurrent adversary and the rushing adversary. They further improve the result resisting an erasure-free adversary for signature schemes and encryption scheme [22], in which (1) an adversary corrupts servers at any point of the protocol depending on its entire view of the computation; (2) when corrupted, the players have to hand over the adversary their entire computation history; i.e., nothing can be erased.

The first distributed key generation of discrete logarithm problem based threshold cryptosystems is proposed in [26] by Pedersen, but later in [16] a flaw of their scheme has been found by Gennaro et al. that the key is not uniformly generated in the key space with some adversary and also a solution is given to deal with such type of adversary. In [8], Canetti et al. improve their results to resist adaptive adversary.

Fouque and Stern has noticed that a one-round key sharing protocol [14] can make the adaptive adversary meaningless, in the sense that it achieves nothing more than what a static adversary achieves. They have equipped a non-interactive proof of verifiable encryption discrete logarithm in the Paillier cryptosystem [25] in place of a complaining phase of key generation. However, in their scheme one can do nothing but choose Paillier cryptosystem as the only tool to construct the secret channel. On the other hand, to distribute discrete logarithm problem based key shares, one must manage keys of extra Paillier

cryptosystems, which may be based on composite degree residuosity assumptions: these degrade the security of the whole system by introducing with new keys of public key cryptosystems under different mathematical assumptions, resulting in a more challenging key management.

1.2 Our Results

We shall focus on one round distributed key generation of discrete logarithm based threshold schemes. Compared to previous schemes, our scheme enjoys following features:

Optimistic execution. First, we note that the servers should not always malfunction. In fact, in a real implementation, we can expect a much optimistic scenario. Previous solutions often make use of both broadcast channels and private channels in sharing phase, however, we notice that a private channel hides the adversary in such a way that if there should arouse a dispute a third party cannot tell whether the sender or the receiver is lying. So a special complaining phase is needed to correctly share a secret without a trusted dealer [26].

Theoretical results in [18,5,28] have already shown the possibility that given fully connected pair-wise secure channels, any multiparty computation (including the distributed key generation) can be performed provided that the number of dishonest players $t < n/3$ and given a broadcast channel together with pairwise private channels, can be secure against a static adversary corrupts up to $t < n/2$ players. However, complexity of their protocols prevents their application in practical use. Besides, since we are to construct a public key scheme, a private channel that guarantees information-theoretic secrecy is too luxury. A broadcast channel plus a public key encryption cryptosystem with "checkability" seems enough for this task. The concurrent adversary and adaptive erasure-free adversary as defined by [21] is of no significance in a one-round non-interactive protocol and a rushing attack will make no harm if we take the advantage of a synchronous network.

Robustness and efficiency. Our protocol provides security against dishonest majority cheaters, namely $t < n$, because the cheaters can be identified and eliminated in the end, while previous multiple-round protocols can work only with $t < n/2$ in the key generation phase thus in the whole protocol. Moreover, there is only a broadcast channel necessary while both broadcast channel and pairwise secret channel are used in [26,8]. We emphasize on the efficiency of our protocol. There are totally n^2 shares and each user broadcasts $O(ln)$ data where l is the security parameter, the whole data flow over the network is $O(ln^2)$. Considering the hidden coefficient in previous schemes and multiple rounds of communication, the complexity is acceptable for a network with low density and it can be further improved in an optimistic sense by using batch encryption techniques. A basic hypothesis here is that in practice a decryption server which stores the secret key share can process a large amount of parallel computations while the communication costs can be neglected.

User preference and flexibility on keys. Since each server may have already set up their encryption keys, introduction of additional Paillier cryptosystem may increase the amount of secret keys in the whole system. Our scheme works best for a situation that every player has his preference on keys. In previous schemes, if the public-secret key pairs of Paillier encryption have not been set up, there must be another auxiliary protocol executed to do the overhead. In our scheme, users have more flexibility.

Generalized design techniques. Another contribution of this work is a generalized template for designing non-interactive protocols. If a multiparty protocol can be simulated as information transmitted from a trusted party to all players then it can be designed using our template. The technique introduced here can be applied to other multiparty protocols as building blocks.

Smallest set of assumptions. A theoretical result of our protocol actually implies that a distributed discrete logarithm key generation protocol can only be based on discrete logarithm problem and existence of a broadcast channel, while previous protocols rely more assumptions beyond, like additional pairwise private channels [26] or other mathematical problem, e.g., deterministic residuosity [25, 14].

2 Preliminary

2.1 Basic Models

There are mainly two types of threshold cryptosystems in practice, namely, those based on the RSA problem and those on the discrete logarithm problem. We shall limit our scope to threshold cryptosystems based on intractability of the discrete logarithm problem because RSA based distributed key generation cannot be made one-round since a primality test is necessary for secret prime factors of public modulus, while discrete logarithm type is more amiable for the selection of secret keys. Let p and q denote large primes such that q divides $p-1$, Z_q is a subgroup of Z_p^* of order q, the secret key is an element x, such that public key is $y = g^x \bmod p$. Also in self-evident context, we also write $x = \log_g^y \bmod p$ as $x = \mathrm{DL}(y)$.

Network and players. We shall assume a synchronous network in all our discussion. Synchronous network can be constructed with GPS (Global Positioning System) [20] in practice. The players in the protocol should be connected with a broadcast channel which sends the same information to every player. There is no need to maintain any pairwise secure channels.

2.2 Security Definitions

Distributed secret key generation of threshold schemes. Assume n players over a synchronous open network are fully connected. A (t, n) distributed

threshold secret key generation is a protocol run among these players, who each chooses a secret number and shares it among all the players. Finally, honest players can decide a unique public key and corresponding secret key shares, such that if $t + 1$ or more players decrypt a ciphertext or issue signatures jointly, while t or less players can computationally have no information on the secret keys assuming that the discrete logarithm problem is intractable.

Possible adversarial behaviors. We follow [22] to describe the capabilities of the adversary. We assume the existence of n parties communicating over a synchronous broadcast channel with rushing, where up to a threshold l parties may be corrupted. Lysyanskaya et al in [22] describe a scheme to deal with such adversary, however, their protocol require interaction in the *Share* phase. An adaptive adversary can depend on the protocol history to schedule his attack while static adversary cannot. An *adaptive* adversary makes no difference from a static adversary if the protocol has only one round. Moreover, the underlying encryption scheme used here need not have chosen ciphertext security. Throughout this paper, the adversary is considered to be passive, whose attack strategy can be assumed to be fixed at the beginning of the protocol.

In our protocol, we permit any adversarial behavior trying to deviate the protocol from its correct execution, namely, an adversary may do any of the following:

- The adversary chooses to at most t players to corrupt. Once corrupted, the players have to hand all their data to the adversary and all their actions are controlled by the adversary.
- Each player chooses a random number and shares it using a public verifiable encryption via broadcast channel, while corrupted player may response correctly or not.
- The adversary can launch pre-scheduled attack during the execution, make arbitrary deviations from the protocol.

Security of distributed secret key generation for threshold cryptosystem. The security of the distributed secret key generation protocol is defined in terms of *correctness* and *confidentiality*. By correctness, we mean the following:

1. Each subset of at least $t + 1$ shares defines the same secret key x. t is called the threshold.
2. All honest parties have the same public key $y = g^x$. This should not be disturbed by a malicious adversary.
3. The secret key is uniformly distributed in the key space. That is, every element of subgroup will be chosen as the secret key with equivalent probability.

Confidentiality property requires that any PPT adversary should not learn any information on the secret key by corrupting at most t players. Furthermore, this can be expressed by simulatability. If a PPT adversary corrupts at most t servers which is erasure-free during the key generation, then the view of the adversary in the real execution shouldn't be distinguishable from the output of a program called a simulator, which is executed in expected polynomial time.

2.3 Homomorphic Encryption Schemes

Definition 1 (Public key encryption). A public key encryption scheme is a 3-tuple algorithm: (K, E, D), where K is the probabilistic key generation algorithm; E, maybe probabilistic, is the encryption algorithm which maps a plaintext m to a ciphertext c; D is a deterministic algorithm, which on input c recovers m.

Definition 2 (Homomorphic encryption). A function $F : G \to H$ maps from group G to group H is called a *homomorphism* of G into H if F preserves the operation of G. If $*$ and $\circ$ are polynomial time computable operations of G and H, respectively, then F preserves the operation of G if for all $a, b \in G$: $F(a * b) = F(a) \circ E(b)$. If an encryption scheme maps plaintext and ciphertext as above relations, we call it a homomorphic encryption scheme.

Remark 1. A homomorphic encryption must be malleable, which will not be IND-CCA secure [3]: given $E(M)$, the adversary can compute easily $E(M*k)$. We emphasize here that we are designing a non-interactive protocol, all encryption will be only be performed once, then discussions on adaptive security (chosen ciphertext security) can be omitted.

3 Previous Schemes

In this section, we briefly review some related protocols designed for distributed discrete logarithm key generation.

3.1 Pedersen's Scheme

Pedersen's scheme was the first attempt to eliminate the trusted dealer to construct a threshold key cryptosystem with shares generated from Shamir's secret sharing scheme [30]. He ingeniously used the homomorphic property of discrete logarithm. Each player acts as the dealer to use a broadcast channel to distribute his public key shares and public checksum and pair-wise private channels are used to distribute the secret share, finally the broadcast channel is used again to identify possible misbehavior.

A player P_i chooses $x_i \in_R Z_q$ at random. He also picks t random numbers $a_{i,k}$ in Z_q. For $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k$ where $a_{i,0} = x_i$, he privately sends a secret share $s_{i,j} = f_i(j) \bmod q$ to another player P_j with the private channel and broadcasts public information $y_{i,j} (= g^{s_{i,j}} \bmod p)$ and $A_{i,k} (= g^{a_{i,k}} \bmod p)$ for $k = 0, 1, .., t$. Each player P_j receives his secret share and the signature of P_i on the share and check if:

$$y_{i,j} = \prod_{k=0}^{t} A_{i,k}^{j^k} \bmod p$$

If $s_{i,j}$ is not the discrete logarithm of $y_{i,j}$ participant P_j broadcasts a complaint against P_i. If more than $n/2$ players complain against player P_i, then it is clear that player P_i is cheating and he will be disqualified. Otherwise, if less than $n/2$ players complained, P_i will reveal the secret key share $s_{i,j}$ and the signature on it for player P_j who has made the complaint. If he fails to reply any of the equation $y_{i,j} = g^{s_{i,j}}$, he will be disqualified. Otherwise he can still be qualified. Later a set of qualified players Q can be decided. The public key will be $y = \prod_{i \in Q} y_i$, where $y_i = A_{i,0} = y^{x_i}$.

Since after the complaining phase revealment of the secret key shares are performed, a flaw was indicated in [16] that the adversary can create a bias in the distribution of the last bit of the public key with probability 3/4 rather than 1/2. A second complaining phase is necessary for the identification of the cheaters and all players have to manage both a broadcast channel and private channels. However, this scheme is simple, no need to maintain other secret key of different cryptosystems.

3.2 Fouque and Stern's Scheme

By noticing the aftereffect of the private channel, Fouque and Stern propose a scheme using only public channel [14]: p'_j, q'_j are large primes and $N_j = p'_j q'_j$ is the modulus in Paillier cryptosystem. G_j is an element in $Z^*_{n^2}$of order a multiple of N_j. Let λ to be the Carmichael lambda fucntion $\lambda(N)$. The public key is $PK_j = (N, G)$ and the secret key is $SK_j = \lambda$.

For a non-interactive proof of fair encryption of discrete logarithm, Player P_i generates a random secret $s_{i,0}$, sets $a_{i,0} = s_{i,0}$ and chooses $a_{i,k}$ at random from Z_q for $1 \leq k \leq t$. Number $a_{i,0}, ..., a_{i,t}$ jointly define the polynomial $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k \in Z_p[X]$. Then he computes $s_{i,j} = f_i(j) \bmod p$. He broadcasts: for $k = 0, ..., t, A_{i,k} = g^{a_{i,k}} \bmod p$ and $y_{i,j} = g^{s_{i,j}} \bmod p$, $Y_{i,j} = G^{s_{i,j}} u_{i,j}^{N_j} \bmod N_j^2$, and a proof $(e_{i,j}, z_{i,j}, w_{i,j}) \in [0, B[\times[0, A[\times[0, N_j[$, where $e_{i,j} = H(g, G, y_{i,j}, Y_{i,j}, g^r y_{i,j}^{-e_{i,j}} \bmod p, G^{z_{i,j}} w^{N_j} Y_{i,j}^{-e_{i,j}} \bmod N_j^2)$, and $y_{i,j}^q = 1 \bmod p$, where $|A| \geq |B| \circ |S| + k$, k is the security parameter.

For each $1 \leq i, j \leq n$, the players verify that:

$$\prod_{k=0}^{t} A_{i,k}^{j^k} = \prod_{k=0}^{t} g^{\sum_{k=o}^{t} a_{i,k} j^k} = g^{f_i(j)} \bmod p$$

and check whether $g^{f_i(j)} \bmod p$ is equal to $y_{i,j}$ in order to verify that the distribution is correct. The players also verify the proofs $(e_{i,j}, w_{i,j}, z_{i,j})$: if $e = H(g, G, y_{i,j}, Y_{i,j}, g^{z_{i,j}} g^{-e_{i,j}} \bmod p, G^{z_{i,j}} w^{N_j} Y_{i,j}^{-e_{i,j}} \bmod N_j^2)$, and $y_{i,j}^p = 1 \bmod p$ for $1 \leq i, j \leq n$.

The players which do not follow the protocol are disqualified. The set of qualified players decide the public key y in the same way as in [26]. And the players will have correct secret key shares distributed. The public key will be the computed using Lagrangian interpolation polynomial. Thus secret key shares of discrete logarithm has been generated for a (t, n) threshold cryptosystem.

Remark 2. In fact, the proof system used in Fouque and Stern scheme based on Paillier encryption scheme is a specially "good" kind of homomorphic encryption schemes where $E(M_1 + M_2) = E(M_1) \times E(M_2)$ and $E(M_1 \times k) = E(M_1)^k$. An earlier work can even be found in Schnorr signature scheme [29]. We argue that e.g. the Nacacche-Stern encryption scheme and Okamoto-Uchiyama encryption scheme holding similar property and proof system can be formed analogously. However, not all the encryption schemes have such ideal properties, which may appear likely to be the following (e.g. RSA): $E(M_1 \times M_2) = E(M_1) \times E(M_2)$ *and* $E(M_1^k) = E(M_1)^k$.

4 Publicly Verifiable Encryption with Proof of Fairness

4.1 A One-Round Model for Threshold Cryptosystem

A trick is played here: a tag is attached to the ciphertext which allows everyone be able to check the validity of the ciphertext without decrypting it. Via a public channel, each player broadcasts a public commitment and verifiable encryption of all his shares. The secret key share to player P_j is encrypted under P_j's public key together with a tag so that everyone may check the correctness of the secret share while having no idea of what it is. So a cheating dealer can be detected and disqualified immediately in the key generation phase. If there is one honest player, the resulting public key and secret key will be distributed uniformly in the key space.

The construction of such a tag can in fact using general zero-knowledge for $\mathcal{NP}$ language combined with any encryption scheme to provide the encrypted plaintext is in fact the discrete logarithm of a publicly known value. However, to emphasize the efficiency of practical use, we construct a verifiable encryption of discrete logarithm over any homomorphic encryption schemes. Note that our definition of homomorphism is much broader.

4.2 Publicly Verifiable Encryption of Discrete Logarithm

The key idea of our publicly verifiable encryption is in fact a non-interactive publicly verifiable encryption, where with a public input y a prover proves a ciphertext c encrypts x under a public key PK (typically belongs to a third party), such that $(x, y) \in R$, where R is a binary relation varying from applications. In the discrete logarithm key generation, R refers to the discrete logarithm, i.e., x is the discrete logarithm of a public value y to the base g. Our protocol differs from the one in [7] is that our underlying encryption scheme is homomorphic, while the one in [7], used in a verifiable signature escrow context, should avoid homomorphic property to maintain chosen ciphertext security. Our scheme apparently may be more efficient than theirs. Abe in [1] has also designed checkable encryption schemes based on various underlying encryption in the random oracle model. However, the construction there is aiming at chosen ciphertext security and actually only proves whether the ciphertext sender knows the corresponding

plaintext. In our key generation protocol, we require not only that the ciphertext is formed validly, but also it should be discrete logarithm of some publicly known value, in other words, the relation R should be verified at the same time. Let's fist look at some examples with typical encryption schemes:

Example 1: ElGamal encryption. This scheme is observed to first appear in [32]. It is a combination of ElGamal encryption scheme and Schnorr signature assuming double exponent exists in a subgroup of prime order. The system assumes to be secure if DDH problem is hard in the subgroup. Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that in (p, q, g) the DDH problem is hard. Here g is a generator of the group defined by p of prime order q.

- **Key generation**: $(p, q, h, z) \leftarrow \mathcal{G}(1^k)$, where h is a generator of Z_q in Z_q^*, $z \in_R Z_p$. Then the secret key is z and the public key is $v(= h^z \bmod p), p, q$ and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ (l is the security parameter, where 2^{-l} is negligible).
- **Encryption**: To verifiably encrypt x satisfying $y = g^x$, set $C = (A, B)$ as standard ElGamal encryption, where $a \in_R Z_q$, $A = h^a$, $B = x^{-1} \cdot v^a$, a proof of fairness can be constructed as follows: select $r_1, ..., r_l \in_R Z_q$, for $1 \leq i \leq l$, compute $w_{1i} = h^{r_i} \bmod p$ and $w_{2i} = g^{v^{r_i}}$. Denote $c = H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$.
- **Validity test**: Verifier checks if $c \stackrel{?}{=} H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$. Let c_i denotes the ith bit of $c = H(v, h, g, p, q, A, B, w_{11}, w_{21}, ..., w_{1l}, w_{2l})$, then for $1 \leq i \leq l$, $b_i = r_i - c_i \cdot a$, check if $w_{i1} \stackrel{?}{=} h^{b_i} A^{c_i}$, and $w_{i2} \stackrel{?}{=} g^{(v^{b_i})}$ if $c_i = 0$, or $w_{i2} \stackrel{?}{=} y^{(B \cdot v^{r_i})}$ if $c_i = 1$.
- **Decryption**: For a valid ciphertext that passes the above test, output $x = A^z / B$ as plaintext.

Example 2: RSA encryption. RSA encryption operates in the group Z_N^*, where $N = p'q'$, p' and q' are large primes. It is trivial that p', q', p, q must be distinct to maintain the system secure.

- **Key generation**: $(p', q') \leftarrow \mathcal{G}(1^k)$, $N = p'q'$, and $\phi(N) = (p' - 1)(q' - 1)$, so that Z_p is a subgroup of Z_N^* and Z_q is a subgroup of Z_p^* of order q. e is a small prime and select private key $d = e^{-1} \bmod \phi(N)$. Publish public key as e, N and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ (l is the security parameter, where 2^{-l} is negligible) as public encryption key.
- **Encryption**: $B = x^e \bmod N$, $y = g^x$, a proof of fairness can be constructed as follows: $r_1, ..., r_l \in_R Z_q$, for $0 \leq i \leq l$, $A_i = r_i^e \bmod N$. let c_i denote ith bit of $c = H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$, $b_i = x^{c_i} \cdot r_l \bmod N$. Then the ciphertext is B and the proof is $(A_1, ..., A_l, b_1, ..., b_l)$.
- **Validity test**: First check if $c \stackrel{?}{=} H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$. For $1 \leq i \leq l$, let c_i denote the ith bit of $c = H(e, N, B, A_1, A_2, ..., A_l, b_1, ..., b_l)$, verify $b_i^e \stackrel{?}{=} A_i$ if $c_i = 0$, or $b_i = B \cdot A_i$ if $c_i = 1$.

– **Decryption**: For a valid ciphertext that passes the above test, output $x = c^d \bmod N$ as plaintext.

Remark 3. In these schemes the total data size broadcast expands with a factor of l, the security parameter, however, the probability error is exponentially small (2^{-l}) in these construction. Note these are not to perform frequently.

5 Generalization to Arbitrary Homomorphic Encryption

In this section, we show the above results can be generalized to arbitrary homomorphic encryption / decryption schemes. As shown above, any encryption scheme that maps a plaintext from the input domain G to a ciphertext in the output domain H preserving operations can be expressed as:

$$E(M_1 * M_2) = E(M_1) \circ E(M_2)$$

Without loss of generality, assume $*$, $\circ$, $\star$ are three operations (e.g. $+$, $\times$, exponentiation in finite field) defined in ascending order of priority in both G and H. For a homomorphic encryption maps preserved operations in G and in H, the homomorphic properties can be written as:

$$* \rightarrow \circ \tag{1}$$
$$\circ \rightarrow \star \tag{2}$$
$$\circ \rightarrow \circ \tag{3}$$
$$\star \rightarrow \star \tag{4}$$

For example, (1) shows that $*$ operation in G (plaintext space) is preserved as $*$ in H (ciphertext space). Since encryption scheme must at least be oneway for it must be infeasible for anyone to infer the ciphertext to the plaintext, then above mappings must hold oneway, which infers that it is possible to open some masked value on the right side while maintaining secret on the left side. For the relation to prove and underlying encryption scheme: either the same operation preserved in G and H, or an operation in G by relation $\mathcal{R}$ is preserved in H by the encryption scheme. If two such pairs can be found to hold for the relation and the encryption scheme, a proof system can then be constructed.

Case 1: The same operation is preserved in the domain for the relation $\mathcal{R}$ and the encryption E, e.g., the pair (1,2) for which $\circ$ is preserved in both G and H holds for both the relation and the homomorphic encryption scheme. The proof system can be constructed efficiently using Fiat-Shamir heuristic [13]. In such case, a challenge of the first homomorphism relation can be easily turned to the second one, then a zero-knowledge proof can be constructed proving the $R(x, y)$ under PK.

Case 2: The same operation is preserved in different domains G and H respectively. E.g. (1,2) for the relation and (3,4) for the homomorphic encryption,

which form the proof system for verifiable RSA encryption of discrete logarithm in Example 2. In such case since there is no direct way bridging over the challenge of the first one to the second one, however, a cut-and-choose methodology can be executed sequentially by mapping (1) (with respect to (2)) in G to (3) (with respect to (4)) in H. In such cases, the error probability is exponentially small after sequential repeated executions.

Generally, we have the following theorem on Case 1:

Theorem 1. *Proofs from Case 1 based on homomorphic encryption can be made zero-knowledge efficiently.*

Proof. We shall first make the construction and then prove that it satisfies the requirements of zero-knowledge. For definitions and precise model of zero-knowledge proof please refer to [17]. There are three operations in all. For relation $\mathcal{R}$, assume $M_1 * M_2 \to M_1 \circ M_2$, $M_1 \circ k \to M_1 \star k$ and homomorphic encryption schemes: $E(M_1 * M_2) = E(M_1) \circ E(M_2)$ and $E(M_1 \circ k) = E(M_1) \star k$. The prover chooses a random element r from G and compute $E(r)$ in H as a commitment. Such commitment by the encryption scheme is unconditionally binding (guaranteed by deterministic property of a public key encryption scheme) and computationally concealing. Then from a public random source (for simplicity, consider a random oracle), a random reference c is derived and the prover uses the two pairs of relations, $E(r * x) \to E(r) \circ E(x)$ and $E(c \circ x) \to E(x) \star c$. He then sends $w = r * x \circ c$, $E(r)$, $E(x)$ to the verifier, who verifies the two pairs of relations: by $E(w) = E(r) \circ E(x) \star c$.

First, it is easily seen that an honest prover will always be able to construct such proof. Second, assume that a cheating prover doesn't know x and is able to compute such proofs. Since c is a public random source accessible by both the prover and verifier, in other words, c is fixed once chosen, then having the ability to find $x' \neq x$ from the output $E(r * x' \circ c) = E(r * x \circ c)$ contradicts the deterministic property of the underlying assumption of the encryption schemes. Third, given w and $E(x)$, the verifier can easily simulates all the information gained in the protocol, by obtaining a random c' from the random public accessible source, calculate x' from $w' = r' * x' \circ c'$ and construct $E(r' * x' \circ c') = E(r') \circ E(x') \star c'$.

Case 2 is a little cumbersome, since no direct operation is preserved in both encryption scheme and fairness of an relation (here discrete logarithm of a public value). Now the relation (here for example discrete logarithm) maps $M_1 * M_2 \to M_1 \circ M_2$, however, the homomorphic encryption only maps $E(M_1 \circ M_2)$ to $E(M_1) \circ E(M_2)$ and $E(M_1 \star k)$ to $E(M_1) \star k$. Since there is no direct method to obtain $E(M_1) \star k$ in general, we cannot proceed except for some special cases where an efficient operation can be achieved for one bit (e.g. homomorphic relation $M \star 0 \to 1$ or $M \star 1 \to M$ hold for discrete logarithm), we can then base the proof on cut-and-choose methodology.

For $1 \leq i \leq l$, the prover chooses r_i randomly from G. Let the public random source output a challenge bit c_i. The prover computes $E(r_i)$ and $w_i = r_i * x \circ c_i$, and together with $E(x)$ sends them to the verifier. The verifier opens every commitment by verifying $E(w_i) = E(r_i) \circ E(x) \star 0 = E(r_i)$, if $c_i = 0$;

$E(w_i) = E(r_i) \circ E(x) \star 1 = E(r_i) \circ E(x)$ if $c_i = 1$. Fortunately, most of finite field arithmetics satisfy this principle. We base our construction on the following theorem:

Theorem 2. *The construction for Case 2 is a secure zero-knowledge proof with error probability exponentially small.*

Proof. Again, the commitment by using public key encryption is again unconditionally binding and computationally concealing. First, completeness is clear that an honest prover can construct such proof. Second, this is a standard "cut-and-choose" argument, that an honest verifier can only be cheated is roughly 2^{-l}, if the l-bit string is chosen uniformly from the public random source. Third, for $1 \leq i \leq l$, let the verifier constructs a simulator outputs random r'_i and forms the proof in the same way $E(r'_i * x \circ c'_i)$, where c'_i is the ith random bit obtained from the public random source. In the view of an adversary whether this is from an honest prover or the simulator is computationally indistinguishable, if $E(r_i)$ and $E(r'_i)$ are identically distributed in H.

Since RSA encryption has the homomorphic encryption property of Case 2, we can further prove its security with similar reasoning:

Corollary 1. *The protocol given in Example 2 is a secure verifiable encryption with proof of fairness, where public randomness is generated by random oracle.*

6 Distributed Key Generation for Discrete Logarithm

6.1 The Scheme

The main protocol uses homomorphic encryption with public verifiable proof of fairness as a building block to build a (t, n) threshold cryptosystem. To have public key shares pk_i of player P_i and corresponding secret key shares $(sk_1, ..., sk_n)$ correctly distributed, suppose each player P_j has his personal public key and secret key pair (E_{PK_j}, D_{SK_j}) (they can be chosen independently and based on different assumptions). Each share is verifiably encrypted and broadcasted to every player so that every player can verify if other players have received the correct share. A cheating dealer can be caught immediately once he cheats. Since the network is supposed to be synchronous, then rushing attacks and other attacks will not take place. The main protocol is performed as follows:

1. For $1 \leq i \leq n$, P_i chooses a random number x_i from the $x_i \in_R Z_p$, computes $y_i = g^{x_i}$, and share x_i with Shamir's secret sharing scheme: he sets: $a_{i,0} = s_{i,0} = x_i$ and chooses $a_{i,k}$ at random from Z_q for $1 \leq k \leq t$, which the number $a_{i,0}, ..., a_{i,t}$ define a polynomial $f_i(X) = \sum_{k=0}^{t} a_{i,k} X^k \in Z_q[X]$ of degree t. Then he computes $s_{i,j} = f_i(j) \bmod p$. He broadcasts: for $k = 0, ..., t$, $A_{i,k} = g^{a_{i,k}} \bmod p$ and $y_{i,j} = g^{s_{i,j}} \bmod p$, and an encryption $E_{PK_j}(s_{i,j})$ of secret key share for Player P_j under the correct publicly verifiable encryption with proof of fairness introduced in section 5. Especially he will keep the share when $j = i$.

2. Player $P_1, ..., P_n$ each verifies

$$\prod_{k=0}^{t} A_{i,k}^{j^k} = \prod_{k=0}^{t} (g^{a_{i,k}})^{j^k} = g^{\sum_{k=o}^{t} a_{i,k} j^k} = g^{f_i(j)}$$

and check whether $g^{f_i(j)}$ is equal to $y_{i,j}$ in order to verify that the distribution is correct. The players also verify the proofs that $E_{PK_j}(s_{i,j})$ is the correct encryption of $s_{i,j}$ to the public key PK_j.

3. The players which do not follow the protocol will be disqualified. The remained set of player forms the set Q and they can generate the threshold key system with the public key $(y, y_1, ..., y_n)$ and secret shares $(x_1, ..., x_n)$ where Player P_j will get correct shares from $i \in$ Q, and compute the following:

$$x_j = \sum_{j \in Q} s_{i,j}, \;\; y_j = g^{x_j}, \;\; y = \prod_{i \in Q} A_{i,0} = g^{\sum_{i \in Q} f_i(0)} \qquad i, j \in \mathrm{Q}$$

respectively, $A_{i,0}$ can be computed from the Lagrangian interpolation polynomial given $A_{i,j}$ $(1 \leq j \leq t+1)$.

4. The secret can be constructed if $t+1$ honest players gather. $sk = \sum_{j=1}^{t+1} b_i x_i$, where $b_i = \prod_{1 \leq k \leq t+1, k \neq j} \frac{j}{k-j}$.

6.2 Security Analysis

We get the following theorem on analysis of our protocol:

Theorem 3. *Our scheme is a secure distributed key generation protocol for a threshold cryptosystem.*

Proof. We build our proof on the following two claims for two requirements in the security definitions:

Claim. Our protocol is correct against passive adversary.

It is clear that any subset of $t+1$ honest players can reconstruct the secret key. For the public verifiability, the honest players can be decided, thus the public key can be uniquely decided. If there should be one honest player, the public key y and the secret key x will be uniformly random in the subgroup.

Claim. Our protocol protects the confidentiality of the secret key.

For the zero-knowledge property of the publicly verifiable encryption scheme, a PPT adversary cannot break the confidentiality of the secret shares. Furthermore, We can build a simulator which is a pre-decided program in stead of real protocol run. In the view of an adversary, the sharing phase can simulated: it is computationally indistinguishable for the adversary to tell the output from a simulator from a real program. Since the verifiable encryption is simulatable.

Combining two claims, we complete the proof of the theorem.

Remark 4. Furthermore, the scheme in [14] can be regarded as a special case if we take the proof knowledge of the style in the Pailler encryption for Case 1.

7 Conclusion

Our key generation protocol is round optimal (one round). Our non-interactive encryption with proof of fairness can be used as building blocks for other complicated schemes. We argue that actually, the technique developed here can be used to any scenario where data flow from the Trusted Authority to the players without interactions. Since homomorphic encryption is malleable, it is not possibly to achieve security against an adaptive adversary under multiple use. However, it is possible to be made *composable* if an independent set up with randomly chosen parameters is applied every time.

References

1. M. Abe. Securing "encryption + proof of knowledge" in the random oracle model. In *CT-RSA 2002*, volume 2271 of *LNCS*, pages 277–289. Springer-Verlag, 2002.
2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, volume 1462 of *LNCS*. Spring-Verlag, 1998.
4. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
5. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC'88. ACM, 1988.
6. C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security (CCS)*, pages 120–127. ACM, 1999.
7. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *ASIACRYPT'00*, volume 1976 of *LNCS*, pages 331–345. Springer-Verlag, 2000.
8. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer-Verlag, 99.
9. R. Cramer, M. Franklin, B. Schoenmakers, , and M. Yung. Multi-authority secret ballot elections with linear work. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 72–83. Springer-Verlag, 1999.
10. Y. Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO'87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1987.
11. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1990.
12. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
13. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problem. In *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1987.

14. P. Fouque and J. Stern. One round threshold discrete-log key generation without private channels. In *PKC'01*, volume 1992 of *LNCS*, pages 300–316. Springer-Verlag, 2001.
15. M. Franklin and M. Reiter. Verifiable signature sharing. In *EUROCRYPT'95*, volume 921 of *LNCS*, pages 50–63. Springer-Verlag, 1995.
16. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.
17. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.
18. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In *19th STOC*, pages 25–27. Springer-Verlag, 1987.
19. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 539–556. Springer-Verlag, 2000.
20. `http://tycho.usno.navy.mil/gps.html`.
21. S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 221–242. Springer-Verlag, 2000.
22. A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *ASIACRYPT'01*, volume 2248 of *LNCS*, pages 331–350. Springer-Verlag, 2001.
23. W. Mao. Verifiable escrowed signature. In *ACISP*, volume 1270 of *LNCS*, pages 240–248. Springer-Verlag, 1997.
24. S. Micali. Fair public-key cryptosystems. In *CRYPTO'92*, volume 740 of *LNCS*, pages 113–138. Springer-Verlag, 1993.
25. P. Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.
26. T.P. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt'91*, volume 547 of *LNCS*, pages 522–526. Springer-Verlag, 1991.
27. B. Pfitzmann, M. Schunter, and M. Waidner. Optimal efficiency of optimistic contract signing. In *17th PODC*, pages 113–122. Springer-Verlag, 1998.
28. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *21th STOC*, LNCS, pages 73–85. Springer-Verlag, 1989.
29. C.P. Schnorr. Ecient signature generation for smart cards. In *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer-Verlag, 1990.
30. A. Shamir. How to share a secret. *Communications of ACM*, 22:612–613, 1979.
31. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attacks. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 1–16. Springer-Verlag, 1998.
32. M. Stadler. Publicly verifiable secret sharing. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer-Verlag, 1996.