抵抗一般结构敌手的自适应安全分布式密钥生成协议*

何云筱+, 李 宝, 吕克伟

(中国科学院 研究生院 信息安全国家重点实验室,北京 100039)

An Adaptively Secure Distributed Key Generation Scheme Against General Adversary without Erasure

HE Yun-Xiao+, LI Bao, LÜ Ke-Wei

(State Key Laboratory of Information Security, Graduate School, The Chinese Academy of Sciences, Beijing 100039, China)

+ Corresponding author: E-mail: heyx@ustc.edu, heyx97@yahoo.com

Received 2003-12-25; Accepted 2004-05-04

He YX, Li B, Lü KW. An adaptively secure distributed key generation scheme against general adversary without erasure. *Journal of Software*, 2005,16(3):453–461. DOI: 10.1360/jos160453

Abstract: Transformation of the widely used Pedersen's Verifiable Secret Sharing (Pedersen-VSS) to Pedersen-VSS-General secure against general adversary is first presented. Then a misunderstanding about the use of zero-knowledge (ZK) proof in the DL-Key-Gen scheme proposed by R. Canetti etc. is pointed out, and an improvement to it is made. An adaptively secure distributed key generation scheme against general adversary without the assumption of erasure is developed. A detailed black-box simulator for the security proof of the proposed scheme is also given.

Key words: distributed key generation; adaptive security; general adversary; verifiable secret sharing; zero-knowledge proof

摘 要: 首先将基于门限结构的彼得森可验证秘密共享方案(Pedersen-VSS)转换成可以抵抗一般结构敌手攻击的方案(Pedersen-VSS-General).指出 R. Canetti 等人在设计分布式密钥生成方案(DL-Key-Gen)时,关于零知识证明使用的一个错误,并给出一种改进方案.基于以上设计,提出一个可以抵御一般结构敌手攻击的自适应安全的分布式密钥生成方案,该方案的安全性不依赖于"擦除"假设.对于这个方案给出详细的基于黑盒模拟的安全性证明.

关键词: 分布式密钥生成;自适应安全;一般结构敌手;可验证秘密共享;零知识证明

中图法分类号: TP309 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.90304013 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA144151 (国家高技术研究发展计划(863)); the Foundation of President of Graduate School of the Chinese Academy of Sciences under Grant No.yzjj2003010 (中国科学院研究生院院长基金)

HE Yun-Xiao was born in 1979. He is a graduate student at Graduate School, the Chinese Academy of Sciences. His current research areas are secure multi-party computation and zero-knowledge proof. **LI Bao** was born in 1962. He is a professor at Graduate School, the Chinese Academy of Sciences. His current research area is information security. **LÜ Ke-Wei** was born in 1970. He is an associate professor at Graduate School, the Chinese Academy of Sciences. His current research areas are secure protocols and proof of security.

1 Introduction

Distributed key generation is a main component of fully distributed cryptosystems. It allows a set of n players to jointly generate a pair of public and private keys without assuming a trusted party. The public key is the output in public, while the private key is maintained as a (i.e. not known to any players) secret shared via a sharing scheme. Each player obtains a share of the private key as his secret that can be later used to enable other distributed cryptosystem such as signature and decryption. This research has attracted a lot of attention in recent years. The first distributed key generation scheme was proposed in Ref.[1]. The basic idea of the protocol (as well as in the subsequent variations (e.g. [2]) is to have n parallel executions of Feldman's verifiable secret sharing protocol (Feldman-VSS)[3] in which each player P_i acts as a dealer of a random secret z_i that he picks. The secret key x is taken to be the sum of the properly shared z_i's. Since Feldman-VSS has the additional property of revealing $y_i = g^{z_i}$, the public key y can be obtained naturally by multiplying all the y_i's that correspond to those properly shared z_i's. This solution has been used in many environments. However in Ref.[4] it is pointed out that Pedersen's scheme can't guarantee the correctness of the output distribution. In the same paper a protocol secure against non-adaptive adversary was proposed by using jointly Feldman-VSS and information-theoretic secure Pedersen-VSS[5]. At the same year, Ref.[6] presented a new protocol secure against adaptive adversary based on Ref.[4] by replacing Feldman-VSS with an honest-verifier ZK proof of knowledge. To ensure that the verifier is honest, Ref.[6] adopts a technique of "distributed challenge generation". By using this technique the assumption of *erasure* is needed to prove the secrecy. This is also considered to achieve high efficiency. We will show in subsection 3.1 that this protocol in fact can't guarantee the claim.

We stress that all results mentioned so far only apply to threshold adversary structures. The contributions include: 1) Transform the widely used Pedersen-VSS to Pedersen-VSS-General secure against general adversary; 2) Present an improvement to the protocol of Ref.[6]. Then based on it we give an adaptively secure distributed key generation protocol (denoted by GDKG) against general adversary; 3) Give a detailed proof of the security via black-box simulation for the protocol. Also we point out that the protocol needs no erasure to prove secrecy when used as an independent scheme.

In Section 2 we present the basic communication and adversarial models, as well as the security requirements and notations for the protocols. In subsection 3.1 we show the misunderstanding in Ref.[6] on using an honest-verifier ZK proof, and introduce a standard technique to transform the ZK proof to one against any verifier. In subsection 3.2 we introduce Monotone Span Programs (MSP) and present the Pedersen-VSS-General. In Section 4 we present the main protocol **GDKG** and in Section 5 we give the detailed proof for its correctness and secrecy.

2 Notation, Model, and Security Requirements

2.1 Notation

Let p, q be two large primes satisfying $q|p-1$, g be an element of order q in Z_p^*, and h be a random element in the subgroup generated by g.

2.2 Model

The participants are a set of n players $P_1,...,P_n$ that can be modeled by PPT(probabilistic polynomial time) Turing machines. We assume a synchronous communication model. All the participants are connected with a complete network of private point-to-point channels. In addition, all players have access to an authenticated broadcast channel. These assumptions allow us to focus on high-level descriptions of the protocols; however, they

may be instantiated using standard cryptographic techniques.

In distributed protocols the adversary is allowed to corrupt any subset of players as specified by the security model (This is modeled by the *Adversary Structure* implemented by an MSP in this paper). A malicious (or Byzantine, active) adversary may cause the participants to deviate arbitrarily from the protocol after corrupting them. An adaptive (or dynamic) adversary means that it can decide which parties to corrupt at any time during the run of the protocol and, in particular, its decisions can be based on the information it gathered during this run. We assume an adaptive and malicious adversary A in this paper. The computational power of the adversary is also modeled by PPT Turing machine.

2.3 Security requirements

In Ref.[4], the requirements of a secure distributed key generation protocol have been introduced for threshold case. Next we will rewrite them for general adversary structure case.

Correctness:

(C1) The shares provided by any subsets of honest players in the access structure (see subsection 4.1) define the same unique secret key x.

(C2) All honest parties have the same value of the public key $y = g^x \pmod p$ where x is the unique secret guaranteed by (C1).

(C3) x is uniformly distributed in Z_q.

Secrecy:

The adversary can learn no additional information on x except for what is implied by the value $y = g^x \pmod p$.

3 Two Main Tools

We introduce two main tools for designing the GDKG protocol in this section.

3.1 Zero-Knowledge proof of knowledge

To prove the knowledge of the discrete-log x of $y = g^x \pmod p$ to the base g, Schnorr presented a 3-round zero-knowledge proof of knowledge[7]:

1. The prover chooses $r \in_R Z_q$, computes $T = g^r$ and sends T to the verifier.
2. The verifier chooses $d \in_R Z_q$ and sends it to the prover.
3. The prover computes $R = r + d \cdot x$ and sends it to the verifier.
4. The verifier checks if $g^R = T \cdot y^d$.

The values T, d and R above are called commitment, challenge and response respectively. It is well known that the above protocol is only *honest-verifier Zero-knowledge*. In Ref.[6] a technique of distributed challenge generation is used to ensure that the verifier is honest i.e. to ensure the challenge d is really random in Z_q. This is considered to achieve *the effect of* $o(n^2)$ *zero-knowledge proofs of knowledge* (*where each of the players proves something to each of the other players*) *in a single* 3-*move honest verifier zero-knowledge proof*. But we stress that although all players participate in the generation of the challenge, only one challenge d has been generated and answered by each player, as a prover. So in fact the error probability of the zero-knowledge proof in Ref.[6] is still $1/q$. This distributed challenge generation results in the dependence on erasure (we will explain this in Section 5).

In this paper we will adopt another standard method to force the verifier to be honest by having the verifier commit to the challenge as a first round. The protocol GDKG performs n parallel zero-knowledge proofs to achieve a more reasonable error probability. Although this may add complexity to the protocol, the elimination of the distributed generation of a challenge has also significantly reduced the complexity, compared to the original protocol in Ref.[6]. We point out that this replacement removes the dependence on erasure.

3.2 Pedersen-VSS-General

3.2.1 LSSS and MSP

In this paper, we consider linear secret sharing schemes (LSSS). An LSSS is defined over a finite field K, and the secret to be distributed is an element in K. Each player receives from the dealer a share consisting of one or more field elements; each share is computed as a fixed linear function of the secret and some random field elements chosen by the dealer. The size of an LSSS is the total number of field elements distributed. Only certain subsets of players, the qualified sets, can reconstruct the secret from their shares. Unqualified sets have no information about the secret. The collection of qualified sets is called the *access structure* of the LSSS (denoted by Γ), and the collection of unqualified sets is called the *adversary structure* (denoted by Δ).

Most proposed secret sharing schemes are linear, but the concept of LSSS was first considered in its full generality by Ref.[8], which introduced the equivalent notion of Monotone Span Programs.

Definition 3.1. The quadrupel $M=(K;M;\varphi;\varepsilon)$ is called a monotone span program, MSP for short, where K is a finite field, M is a matrix (with f rows and $e+1$ columns, $e+1 \le f$) over K, $\varphi:\{1,\ldots,f\}\to\{1,\ldots,n\}$ is a surjective function and $\varepsilon=(1,0,\ldots,0)$ is the target vector (without loss of generality. For a detailed introduction, see Ref.[9]). The size of M is f.

MSP's and LSSS's are in natural 1-1 correspondence[10]. Hence one can identify an LSSS with its underlying MSP. Here we show how to construct a linear secret sharing scheme from an MSP M.

To distribute $s \in K$:

1. The dealer chooses a random vector $b:=(b_0, b_1, \ldots, b_e)$ where $b_0 = s$.
2. The dealer computes and sends $\langle M_l, b\rangle$ to $P_{\varphi(l)}$ for each $l=1,\ldots,f$

About the reconstruction phase, we have the following proposition:

Proposition 3.2. A set of players G can reconstruct s precisely $\Leftrightarrow \varepsilon \in \mathrm{Im}\, M_G^T$. Otherwise they get no information on s (see Ref.[8] for a proof of this).

Definition 3.3. The MSP M (LSSS) is said to compute the access structure Γ, if $G\in\Gamma \Leftrightarrow \varepsilon \in \mathrm{Im}\, M_G^T$ where the matrix M_G consists of the rows of M which are labeled by a number in G. On the other hand, say M (or corresponding LSSS) implements adversary structure Δ, if Δ is the collection of unqualified sets of M (or corresponding LSSS).

3.2.2 Pedersen-VSS-General

In Ref.[5] an information-theoretic secure VSS, denoted by Pedersen-VSS, against threshold adversary structure was proposed. Assume Δ is the adversary structure that our scheme need to deal with and $M=(K;M;\varphi;\varepsilon)$ is an MSP implementing Δ. Now we show how to build Pedersen-VSS-General (see Fig.1) secure against general adversary structure from a given MSP. The resulting protocols will be secure for whatever adversary structure the MSP happens to implement. Also we refer that for the practical interests, Ref.[11] has introduced how to build an MSP secure against a given adversary structure.

We stress that Pedersen-VSS-General retains the main properties of Pedersen-VSS which are summarized as the following lemma:

Lemma 3.3. Pedersen-VSS-General satisfies the following properties in the presence of a PPT adversary that can corrupts the subsets in the adversary structure implemented by the original MSP:

1. If the dealer is not disqualified during the protocol then all honest players hold shares that suffice to efficiently reconstruct the secret *s*.
2. The protocol produces enough information to prevent cheating at reconstruction time.
3. The view of the adversary is independent of the value of the secret *s*.

To distribute $s \in K$ securely:

1. The dealer chooses two random vector $b:=(b_0, b_1, \ldots, b_e)$ where $b_0 = s$ and $b':=(b'_0, b'_1, \ldots, b'_e)$
2. The dealer computes and sends $s_l=\langle M_l, b\rangle$, $s'_l=\langle M_l, b'\rangle$ $s'_l=$ to $P_{\varphi(l)}$ for each $l=0,1,\ldots,f$; The dealer also Computes and broadcasts $C_j= g^{b_j} \cdot h^{b'_j}$, $j=0,1,\ldots,e$.
3. Each player P_i checks if

$$g^{s_l} \cdot h^{s'_l} = \prod_{k=0}^{e} C_k^{M_{lk}} \tag{1}$$

for each l satisfying $\varphi(l)=i$.

If a check fails, P_i broadcasts a complaint (0,dealer, l, i) against the dealer.

4. For each complaint(0,dealer, l, i), the dealer broadcasts the values s_l, s'_l that satisfy Eq.(1).
5. All the players reject the dealer if
   (1) he received complaints from all players of a subset A where $A \notin \Delta$, or
   (2) he answered some complaints in Step 5 with values not satisfying Eq.(1);

Otherwise they accept the dealer.

Fig.1 Pedersen-VSS-General

4 GDKG Protocol

The distributed key generation protocol secure against general adversary without the assumption of erasure is presented in detail in Fig.2. We denote this protocol as **GDKG**. The solution uses a similar structure as the DKG in Ref.[4] and DL-Key-Gen in Ref.[6]. But we use different VSS scheme and ZK proof of knowledge to achieve correctness and secrecy against general adversary. The protocol proceeds as follows:

Generating x (Steps 1–3): This is achieved by having each player commit to a random value z_i via a Joint Pedersen-VSS-General (Step 1). These commitments are verified by other players and the set of parties passing verification is denoted by *QUEL*(Step 2). Then the shared secret is set (implicitly) to $x = \sum_{i \in QUAL} z_i \pmod q$ (Step 3). In addition to enabling the generation of a random、uniformly distributed value x, the n parallel General-Pedersen-VSS for generating x has the side effect of having each player broadcast an information-theoretically private commitment to z_i of the form $C_{i0} = g^{b_{i0}} \cdot h^{b'_{i0}}$ which enables the "extraction" and publication of the public key y next.

Extracting y (Steps 4–10): Because we in fact set $y = g^x = g^{\sum_{i \in QUAL} z_i} = \prod_{i \in QUAL} g^{z_i} \pmod p$, we could compute it if

Input: Parameters (p,q,g) and h, an element in the subgroup generated by g

Public Output: y, the public key

Secret Output of P_i: $\{x_l \mid \varphi(l)=i\}$, the share of the random secret x distributed by the LSSS (all other secret outputs are erased)

Generating x:

1. Each player P_i, as a dealer, performs a Pedersen-VSS-General of a random z_i he picks:

(a) P_i chooses two random vectors $b_i=(b_{i0}, b_{i1}, \ldots, b_{ie})$ where $b_{i0}=z_i$ and $b'_i=(b'_{i0}, b'_{i1}, \ldots, b'_{ie})$. P_i broadcasts $C_{ik}=g^{b_{ik}}h^{b'_{ik}}$, $k=0,1,\ldots,e$. P_i computes the shares $s_{il}=\langle M_l, b_i\rangle$, $s'_{il}=\langle M_l, b'_i\rangle$ (mod q) and sends them to $P_{\varphi(l)}$ for each l=0,1,…,f.

(b) Each player P_j verifies the shares he received from the other players. For each i=1,…,n , and each l satisfying $\varphi(l)=j$, P_j checks if:

$$g^{s_{il}} \cdot h^{s'_{il}} = \prod_{k=0}^{e} (C_{ik})^{M_{lk}} \tag{2}$$

If the check fails for an index (i,l), P_j broadcasts a complaint (0,i,l,j) against P_i.

(c) Each player P_i who, as a dealer, received a complaint (0,i,l,j), broadcasts the values s_{il} , s'_{il} that satisfy Eq.(2).

(d) Each player marks as *disqualified* any player that either

- received complains from all players of a subset A where $A \notin \Delta$, or
- answered to a complaint in Step 1(c) with values that falsify Eq.(2)

2. Each player builds the set of non-disqualified players *QUAL*.

3. Each player P_j sets his share of the secret as the set of $x_l=\sum_{i\in QUAL} s_{il}$ (mod q) and $x'_l=\sum_{i\in QUAL} s'_{il}$ (mod q) for each l,satisfying$\varphi(l)=j$.

Extracting $y=g^x$ (mod p):

Each player P_i exposes $y_i=g^{z_i}$ (mod p) to enable the computation of $y=g^x$ (mod p).

4. Each player P_j chooses a random value d_j, and broadcasts $D_j=g^{d_j}$ (mod p).

5. Each player P_i, $i\in QUEL$, broadcasts $A_i=g^{b_{i0}}=g^{z_i}$ (mod p) and $B_i=h^{b'_{i0}}$ (mod p) s.t. $C_{i0}=A_i \cdot B_i$. P_i also chooses random values r_{ij} , r'_{ij} and sends $T_{ij}=g^{r_{ij}}$, $T'_{ij}=h^{r'_{ij}}$ (mod p) to P_j for j=1,…,n.

6. Each player P_j broadcasts d_j.

7. Each player P_i, $i\in QUEL$, checks for each P_j if $D_j=g^{dj}$ (mod p). For P_j who passes the verification, P_i computes $R_{ij}=r_{ij}+d_j-b_{i0}$, $R'_{ij}=r'_{ij}+d_j-b'_{i0}$ and sends them to P_j.

8. Each player P_j checks for each P_i, $i\in QUEL$, that $g^{R_{ij}}=T_{ij}-A_i^{d_j}$ and $h^{R'_{ij}}=T'_{ij}-B_i^{d_j}$. If the check fails for an index i, P_j broadcasts a complaint (0,i,j) against P_i.

9. If a player P_i, $i\in QUEL$, receives complains from all players of a subset A where $A\notin\Delta$, then each player P_j broadcasts his share from P_i (i.e. all s_{il} satisfying $\varphi(l)=j$).

Reconstruct z_i z_i from (s_{i1},…,s_{id}) and set $A_i=g^{b_{i0}}=g^{z_i}$ (mod p)

10. Each player computes $y=\prod_{i\in QUAl} A_i$ (mod p)

Fig.2 GDKG

We could have each player "deliver" g^{z_i} ($i\in QUEL$) in a verifiable way. To that end, we require each P_i to

"split" his commitment C_{i0} into two components $A_i = g^{z_i} \pmod p$ and $B_i = h^{b'_{i0}} \pmod p$ (Step 5). To show that he gives the correct split, each P_i proves that he knows both $D\log_g A_i$ and $D\log_h B_i$ with the zero-knowledge proof presented in subsection 3.1(Steps 4–8). If a player fails to prove a correct split, then his value z_i is reconstructed (Step 9). Combined with the assumption that the adversary is PPT (this implies that to compute a discrete-log is infeasible), we can be confident that the broadcast of A_i by the players who pass the verification of Step 8 are really the values we need. For the players who don't pass the verification of Step 8, the reconstruction of Step 9 ensures this.

5 Security Proof for GDKG

In this section we prove the protocol GDKG satisfies the security requirements in subsection 2.3.

Correctness of GDKG: The main difference between GDKG and DL-Key-Gen[6] is that we replace Pedersen-VSS and distributed challenge generation in Ref.[6] with Pedersen-VSS-General and first round commitment respectively. As we have pointed out that both of the new tools retain the security of the old. Hence it is easy for the readers to obtain the confidence of correctness from the proof of correctness in Ref.[6](or Ref.[4], because these three protocols enjoy the similar structure). We stress that at the end of GDKG protocol the shared secret key is implicitly $x = \sum_{i \in QUAL} z_i \pmod q$, the public key is $y = g^x \pmod p$ and each player P_i gets his share $\{ x_l \mid \varphi(l) = i,\ l = 0,1,\ldots, f \}$ of x distributed by the MSP M.

Adaptive security of DGKG: The basic idea is to present a *black-box simulator* SIM for the GDKG protocol (see Fig.3). The *black-box simulator* SIM can access a black-box (or oracle) representing the input-output relations of the real-life adversary. The same idea has been widely used in the proof of zero-knowledge (e.g. [12]) and the security of multiparty computation protocols (e.g. [13]). This simulation is the crux of the secrecy proof. Because the success of simulation implies that anything the adversary gets by participating the protocol can be obtained by the SIM itself without interfering in the run of the protocol, we can make sure that no information about the private key x beyond the value $y = g^x \pmod p$ is revealed in the process of the protocol.

To show this we provide the value of y as input to the simulator SIM and require it to simulate a run of GDKG that ends with y as its public output. We denote by G(resp. B) the set of *currently* good (resp. bad) players. The simulator acts according to the protocol GDKG for all the players in G except a special one P. The state of P (selected at random by SIM) will be used by the simulator to "fix" the output of the simulation to y. Because we have transformed the ZK proof in the presence of honest verifier to the one against any verifier, the simulator SIM can still prove that it knows P's contribution (in Steps 4–8 of GDKG) although it doesn't know some secret information relative to this special player (in particular the component z_P that this player contributes to the secret key). However, if the adversary corrupts P during the simulation (with probability which is relative to the adversary structure the MSP implements, but is small enough for the environment) the simulator will not be able to provide the internal state of this player. Thus, the simulator SIM will need to rewind the adversary and select another special player.

For the DL-Key-Gen in Ref.[6], to prove its secrecy the assumption of erasure is needed due to its use of distributed challenge generation. To ensure the correctness i.e. the randomness of the challenge, many commitments have to be used, and this unavoidably makes the difficulty to prove adaptive secrecy. Fortunately, after we replaced

distributed challenge generation with the first round commitment, we need no longer the assumption of erasure when we use GDKG alone. If we use it as a module in a large system, we can achieve secrecy only by requiring each player P_i erases all secret information generated in the protocol aside from his share $\{ x_l \mid \varphi(l) = i \}$ at the end of the Step 10(in fact, even no erasure is executed we meet with difficulty only when the special player P is corrupted).

Input: public key y, parameters (p,q,g), and h

1. Perform Step 1 to Step 3 of GDKG on behalf of the players in G. At the end of this step the set $QUAL$ is defined. Perform the following pre-computations:
   - Choose one honest player $P \in G$ randomly
   - Compute $A_i = g^{b_{i0}} = g^{z_i} \pmod p$ and $B_i = h^{b'_{i0}} \pmod p$ for $i \in QUEL/\{P\}$
   - Set $A_P^* = y - \prod_{i \in QUEL \setminus \{P\}} (A_i)^{-1} \pmod p$ and $B_P^* = C_{P0} / A_P^* \pmod p$
2. For each player P_j, $j \in G$, perform Step 4 of GDKG;

   For each player P_j, $j \in G$, SIM initiates an execution of the black-box which will return the d_i and D_i, $i \in B$, as well as some requests of corruption.

   (Thus SIM knows all d_j's, in particular the corrupted players', and all these values have the same distribution as in the real life run).
3. For each player P_j, $i \in G \setminus \{P\}$, execute Step 5 of GDKG according to the protocol.

   For player P, pick random values $R_P^*, R_P'^* \in {}_R Z_q$, set $T_{Pj}^* = g^{R_{Pj}^*} \cdot (A_P^*)^{-d_j}$ and $T_{Pj}'^* = g^{R'^*_{Pj}} \cdot (B_P^*)^{-d_j}$. Then broadcast A_P^*, B_P^* and send T_{Pj}^*, $T_{Pj}'^*$ to P_j for j=1,…,n.
4. Broadcast d_i for each $i \in G$.
5. For each player P_i, $i \in G \setminus \{P\}$, execute Step 7 of GDKG according to the protocol;

   For player P, check for each P_j that $D_j = g^{d_i} \pmod p$. For P_j who passes the verification, P sends R_P^*, $R_P'^*$ to P_j.
6. Execute Step 8 of GDKG on behalf of the players in G
7. For each player in $QUEL$ who received complaints from a subset A where $A \notin \Delta$, participate in the reconstruction of their values.

Fig.3 SIM—The black box simulator

We have the following theorem about the effect of the simulator SIM.

Theorem 5.1. Simulator SIM on input (p,q,g,h,y) ends in expected polynomial time and computes a view for the adversary that is identical to the view in the protocol GDKG on input $(p,q,g,h,)$ and output y.

Proof. The view of the adversary in GDKG includes two parts: the public view the adversary sees and the internal states of the corrupted parties.

Firstly we show that SIM outputs a probability distribution that is identical to the public view in the real run of GDKG.

1. For Steps 1-3 of GDKG, the simulation of SIM is carried out according to the protocol, thus all the values generated in the simulation have the same distribution as that in the real run.

2. Steps 4-8 of GDKG are the parallel executions of the zero-knowledge proof. Steps 2-8 of SIM have simulated them.

The values d_i for $i \in G$ are picked randomly as in the real protocol.

The values A_i for $i \in G \setminus \{P\}$ are distributed exactly in the real protocol. As for the value $A_P^* =$

$y\cdot\prod_{i\in QUEL\setminus\{P\}}(A_i)^{-1} \pmod p$ and $B_P^* = C_{P0} / A_P^* \pmod p$, **SIM** computes them in the pre-computations according to the same constraints (i.e. its relationship with y and other A_i for $i \in QUEL\setminus\{P\}$) as in the real execution.

The values R_{ij} and R'_{ij} for $i \in G$ and j =1,…,n are uniformly distributed in Z_q.

In step 3 of SIM, the values $T_{ij} = g^{r_{ij}}$, $T'_{ij} = h^{r'_{ij}} \pmod p$ for $i \in G\setminus\{P\}$ and j =1,…,n are picked randomly as in GDKG.

For values $T_{Pj}^* = g^{R_{Pj}^*}\cdot(A_P^*)^{-d_j}$ and $T_{Pj}'^* = g^{R_{Pj}^*}\cdot(B_P^*)^{-d_j}$, the simulator computes them in step 3 of SIM according to the same constraint (i.e. the verification equation in step 8 of GDKG) as in the real execution.

So the public view in the simulation is identical to that in the real run of GDKG.

Next we show the simulator can produce a consistent view of the internal state for the corrupted players. Clearly, if a player is corrupted before step 2 of SIM, the simulator can produce a consistent view because it is following the protocol GDKG. After step 2 the simulator can show correct internal states for all the players in G except for the special player P because the simulator is PPT and has modified some of the P's public states. Thus, if P is corrupted the simulator has to rewind the black-box to the beginning of step 2 and selects at random a different special player. This rewind makes the SIM end in the expected polynomial time, not polynomial time. □

6 Conclusion

In this paper we propose an adaptively secure distributed key generation protocol (GDKG) against general adversary and give a complete proof for its correctness and secrecy via black box simulation. It doesn't need the assumption of erasure, which is usually an unavoidable technique for designing adaptive secure protocol. It also has the corresponding complexity with the DL-Key-Gen in Ref.[6].

References:

[1] Pedersen T. A threshold cryptosystem without a trusted party. In: Proc. of the Eurocrypt'91. LNCS 547, Springer-Verlag, 1991. 522–526.

[2] Gennaro R, Jarecki S, Krawczyk H, Rabin T. Robust threshold DSS signatures. In: Proc. of the Eurocrypt'96. LNCS 1070, Springer-Verlag, 1996. 354–371.

[3] Feldman P. A practical scheme for non-interactive verifiable secret sharing. In: Proc. of the 28th FOCS. 1987. 427–437.

[4] Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems. In: Proc. of the Eurocrypt'99. LNCS 1592, Springer, 1990. 295–310.

[5] Pedersen T. Non-Interactive and information-theoretic secure verifiable secret sharing. In: Proc. of the Crypt'91. LNCS 576, Springer-Verlag, 1991. 129–140.

[6] Canetti R, Gennaro R, Jarecki S, Krawczyk H, Rabin T. Adaptive security for threshold cryptosystems. In: Proc. of the Crypto'99. LNCS 1666, Springer-Verlag, 1999. 98–115.

[7] Schnorr CP. Efficient signature generation by smart cards. Journal of Cryptology, 1991,4(3):161–174.

[8] Karchmer M, Wigderson A. On span programs. In: Proc. of the 8th Annual Structure in Complexity Theory Conf. IEEE Computer Society Press, 1993. 102–111.

[9] Fehr S. Efficient construction of dual Span Program. Manuscript, 1999.

[10] Beimel A. Secure schemes for secret sharing and key distribution [Ph.D. Thesis]. Israel Institute of Technology, 1996.

[11] Cramer R, Damagard I, Maurer U. General secure multi-party computation from any linear secret sharing scheme. In: Proc. of the Eurocrypt2000. LNCS 1807, Springer-Verlag, 2000. 316–334.

[12] Goldreich O. Foundations of Cryptography Basic Tools. Publishing House of Electronics Industry, 2003.

[13] Canetti R. Studies in secure multi-party computation and application. [Ph.D. Thesis]. Weizmann Institute of Science, 1995.