Practical Distributed Key Generation Scheme and Its application

Chen Huiyan[1,2], Li Zichen[1], *et al.*

[1] Beijing Electronic Science and Technology Institute, BEIJING 100070
[2] E-mail: chenhy2003@gmail.com

Abstract. Generating a distributed key, where a constant fraction of the players can reconstruct the key, is an essential component of threshold cryptography. Previous solutions are based on univariate polynomials. In some applications, these solutions don't provide good service. We present a new distributed key generation protocol. The key idea of our scheme is to use bivariate symmetric polynomial. Compared with other solutions, our construction is efficient in computation and simple and flexible in applications. In addition, our construction admits a rigorous proof of security.

Keywords. Threshold cryptography, bivariate polynomials , distributed key generation, discrete logarithm

1 Introduction

1.1 Background and Related Work

In order to provide security to applications that are inherently distributed, namely, several parties are trying to accomplish some common task (e.g., secure elections, auctions, games) in the presence of an attacker, avoid single-point failure in a security system, and so on, threshold cryptosystems have received a lot of attention in modern cryptographic research, e.g., [1, 2, 3, 4]. In many contexts, it is impractical or impossible to assume that a trusted third party, a.k.a a trusted dealer, is present to generate and distribute key shares to users in such systems. Consequently, distributed key generation (DKG) is desirable in principle. In essence, a distributed key generation scheme allows a set of n players to jointly generate a pair of public and private keys according to the distribution defined by the underlying cryptosysytem without assuming the existence of any trusted party. While public key is output in the clear, the private key is shared by the n players, each of which has its own unique secret share, so that any sufficiently large subset of the n players can reveal or use the key. Commonly the security parameter of such a system is called the threshold, t, which is the number of players that can be corrupted without the key being compromised.

Distributed key generation protocols usually fall into two areas : the distribution of a secret for discrete-log based cryptosystems and the distribution of RSA keys. The latter case is partially solved by the nice paper of Boneh and Franklin [20]. However, the protocol does not allow to efficiently share RSA modulus with strong primes and is not robust against cheaters. Following this paper, two articles [21, 22]provide robustness using different techniques. In this work, We shall limit our scope to distributed key generation for discrete-log cryptosystems.

Methods for distributed key generation for discrete-log cryptosystems have been known for a long time, starting a simple polynomial-based scheme presented by Pedersen [5]. Pedersen's scheme was then used in many discrete-log based threshold cryptosystems, e.g., [6, 7, 8]. It seems necessary that the output distribution of distributed key generation schemes must be the same in the threshold and the centralized case if one attempts to argue the security of the threshold cryptosystem by reducing it to the security of the underlying centralized cryptosystem. But later in [9] (extended version in [10]) a flaw of Pedersen's scheme was found by Gennaro *et al.* that the key is not uniformly generated in the key space with a malicious adversary and also a solution is given to deal with such type of adversary. In [11], Canetti *et al.* improve their results to resist a malicious, adaptive adversary. In [12], John Canny *et al.* also proposed a new distributed key generation scheme based on Gennaro *et al.*'s scheme. Besides the above schemes, there are lots of other distributed key generation schemes, for example,[13, 14, 15]. However, these distributed key generation schemes have a common property, i.e, they all are based on univariate polynomials. But, this property has side effect to these distributed key generation schemes. In some practical applications, e.g, [16, 17, 18, 24, 25] , assuming that players $\mathcal{P}_1$, ..., $\mathcal{P}_n$ (their identities will be assigned with their own indices) have jointly generated a pair of private and public keys $< x, y > \in Z_q \times Z_p$ by one of these schemes, where p is a large prime and q a large prime dividing $p-1$, each player $\mathcal{P}_i$ $(i \in \{1, 2, \ldots, n\})$ holds a share S_i of the secret key x, and any $t+1$ players among $\mathcal{P}_1$, ..., $\mathcal{P}_n$ can jointly recover x. When a new good player $\mathcal{P}_{n+1}$ hopes to hold a share of x, we will show that these schemes can not provide a good service. In fact, making $\mathcal{P}_{n+1}$ share x with players $\mathcal{P}_1$, ..., $\mathcal{P}_n$ needs a coalition of at least $t+1$ players among $\mathcal{P}_1$, ..., $\mathcal{P}_n$ by Lagrange interpolation. Without loss of generality, we assume players to be $\mathcal{P}_1$, ..., $\mathcal{P}_{t+1}$. Then, each player $\mathcal{P}_i$, $i \in \{1, 2, \ldots, t+1\}$, computes a partial secret share $S_{i,n+1}$ from its secret share S_i and returns $S_{i,n+1}$ to $\mathcal{P}_{n+1}$. Thus, $\mathcal{P}_{n+1}$ can get a share S_{n+1} of x as follows by the sum of these $t+1$ partial secret shares $S_{i,n+1}$, $i \in \{1, 2, \ldots, t+1\}$.

$$\begin{aligned} S_{n+1} &= \sum_{i=1}^{t+1} S_i l_i(n+1) = \sum_{i=1}^{t+1} S_{i,n+1} \\ l_i(n+1) &= \frac{(n+1-1)\ldots(n+1-(i-1))(n+1-(i+1))\ldots(n+1-(t+1))}{(i-1)\ldots(i-(i-1))((i)-(i+1))\ldots(i-(t+1))}, \end{aligned}$$

Since Lagrange coefficients $l_i(n+1)$, $i \in \{1, 2, \ldots, t+1\}$, are publicly known, $\mathcal{P}_{n+1}$ can derive S_i by $S_{i,n+1}$, and then easily computes x.

The main reason for producing the above problem is because the existing distributed key generation schemes are based on univariate polynomials. In order to solve the above problem, J. Kong *et al.*[19] presented a shuffling scheme. In the shuffling scheme, a random number is exchanged between any two players $\mathcal{P}_i$ and $\mathcal{P}_j$ (where $i, j \in \{1, 2, \ldots, t+1\}$ and $i \neq j$). One of these two players treats this random number as a positive number while the other side treats it as a negative number. Such each $\mathcal{P}_i$ (where $i \in \{1, 2, \ldots, t+1\}$) has t random numbers

N_i, where $i \in \{1, 2, \dots, t\}$. Each $\mathcal{P}_i$ sums $S_{i,n+1}$ and these t random numbers N_1, ..., N_t, respectively, and then sends such a $S^{'}_{i,n+1} = S_{i,n+1} + \sum_{i=1}^{t} N_i$ to $\mathcal{P}_{n+1}$ instead of $S_{i,n+1}$. It is easy to verify that $\mathcal{P}_{n+1}$ obtains the same value S_{n+1}=$\sum_{i=1}^{t+1} S^{'}_{i,n+1}$. Apparently, J. Kong *et al.*'s method makes distributed key generation schemes become more complicated. In this paper, we try to solve this problem by proposing a new distributed key generation protocol.

1.2 Our Contribution

In this paper, we present a distributed key generation scheme based on bivariate symmetric polynomial(or BDKG, for short). Similar to Gennaro *et al.*'scheme [9], BDKG also consists of two phases: in the first phase, the group of non-disqualified players, $\mathcal{Q}$, is selected, and players in $\mathcal{Q}$ jointly generate a random secret key such that all players in $\mathcal{Q}$ have a share of the secret key, and in the second phase, the public value associated to the shared secret is made. The second phase of BDKG is similar to that of Gennaro *et al.*'scheme. However, in the first phase, BDKG is fully different from Gennaro *et al.*'scheme. The security of Gennaro *et al.*'scheme in the first phase is based on the hardness of solving discrete logarithm. BDKG is based on the results of [23] and is unconditionally secure and does not require expensive computations in the first phase. Thus, BDKG is very efficient computationally.

The public key is fixed once it is generated with BDKG . When a good player $\mathcal{P}_x$ wants to hold a share of secret value corresponding to the above public key and become a member of $\mathcal{Q}$, players in $\mathcal{Q}$ can independently provide $\mathcal{P}_x$ with information on the above secret value and need not exchange information (e.g, random numbers,etc.), which is used to protect the privacy of the above secret value, with other players in $\mathcal{Q}$. Thus, BDKG solves the problem in Section 1.1.

In BDKG, our adversary is static, and BDKG is proven secure only against a static adversary.

The paper will proceed as follows. In Section 2 we present the basic communication and adversarial models for our protocol. In Section 3 we present our solution and its full analysis. In Section 4, we simply discuss the applications of BDKG. Section 5 concludes this paper.

2 Preliminaries

2.1 Communication Model

The systems we describe involve a group of n players. The players are modeled by probabilistic polynomial-time Turing machines. They are connected by a complete network of private point-to-point channels. In addition, the players have access to a dedicated broadcast channel. For a broadcast message, it either reaches all recipients or none. Furthermore, if it reaches all recipients, the order for the recipients is random, i.e., the arrival ordering of a message to the players from a given sender is arbitrary. The messages sent on either a point-to-point or the broadcast channel are received by their recipients within some fixed time bound. In addition, we assume that the honest players start a given round of a protocol at the same time.

2.2 The Adversary

We assume that an adversary, $\mathcal{A}$, can corrupt up to t of the n players in the network, for any value of $t < (n-2)/3$. We consider a malicious adversary that may cause corrupted players to divert from the specified protocol in any way.We assume that the computational power of the adversary is adequately modeled by a probabilistic polynomial time Turing machine. Our adversaries are static. Before the protocol executes, this type of adversaries has already decided which player to corrupt during the execution of the protocol. In other words, he can not change to attack another player by exploiting the runtime information obtained during protocol execution.

2.3 Requirements of Secure DKG Protocol

The following definitions apply to discrete-log based cryptosystems. The globally known constants are q, G, P, where q is a large prime; G, over which discrete logarithm problem[3] is hard, is a cycle a cyclic additive group of order q and P is its generator. The first three criteria of the following definition have been used widely to define DKG protocols. The fourth was added by Gennaro, *et al.* in order to quantify the secrecy of an algorithms key against malicious players in the generation phase.

Definition 1. A t-secure distributed key generation algorithm satisfies the following requirements, assuming the set of players controlled by the adversary is less than t:

Correctness:

(C1) A ll subsets of $t+1$ shares provided by honest players define the same unique secret key x.

(C2) A ll honest parties have the same value of the public key $y = xP$, where x is the unique secret guaranteed by (C1).

(C3) x is uniformly distributed in Z_q (and hence y is uniformly distributed in the subgroup generated by P).

Secrecy:

(S1) The adversary can learn no information about x except for what is implied by the value $y = xP$.

[3] Discrete logarithm problem over G which is a cyclic additive group of order q: Given a generator P of G, and $y = xP, x \in Z_q$, compute x.

3 New Distributed Key Generation Scheme

Our construction, BDKG, can work with less than $\frac{n-2}{3}$ corrupted players, because it is based on so-called bivariate symmetrical polynomials which have been employed for related purposes in the literature [27, 28, 29] and the results of [23]. The description of BDKG is based on some group G[4] which is a cyclic additive group of prime order q. In fact, we can also describe BDKG over multiplicative cycle group with a element of prime order q, e.g, Z_p^* where p is a large prime and q divides $(p-1)$.

3.1 Description of BDKG

In BDKG, we assume that the identity of each player is assigned with his own index. EBDKG works as follows.

1. Start with a dealing phase so that all players know q, G, P, where q is a large prime, G is a addition cycle group of prime order q, P is the generator of G.
2. Generating secret value x (this process called as *Joint-VSS*(t)):
   (a) Each player $\mathcal{P}_i$ randomly chooses a bivariate symmetrical polynomial $f_i(x,z)$ over Z_q of degree t :

   $$f_i(x,z) = \sum_{k=0}^{t}\sum_{j=0}^{t} a_{kj}^{(i)} x^k z^j$$

   where $a_{00}^{(i)} = z_i$, $a_{kj}^{(i)} \in Z_q$, the degree of both x and z equals to t, and $a_{kj}^{(i)}{=}a_{jk}^{(i)}$. And then, $\mathcal{P}_i$ computes $h_m^i(x) = f_i(x,m)$ as follows:

   $$h_m^i(x) = f_i(x,m) = \sum_{k=0}^{t}\sum_{j=0}^{t} a_{kj}^{(i)} x^k m^j,$$

   and securely send $h_m^i(x)$ to $\mathcal{P}_m$, here $m = 1,2,\ldots,n$.

   (b) After receiving $h_m^i(x)$, $\mathcal{P}_m$ computes $h_{mk}^i = h_m^i(k)$ and securely sends it to $\mathcal{P}_k$, here $k = 1,2,\ldots,n$.

   (c) For each $m = 1,2,\ldots,n$, player $\mathcal{P}_k$ checks if

   $$h_{mk}^i = h_k^i(m) \tag{1}$$

   If there are at least $t{+}1$ formulas that are not via checking, $\mathcal{P}_k$ broadcasts a complaint against $\mathcal{P}_i$.

   (d) Each player $\mathcal{P}_i$ who receives at most t complaints broadcasts $h_k^i(x)$ that satisfies Eq.1. Each of other $n-2$ players, $\mathcal{P}_j$, checks if $h_k^i(j) = h_j^i(k)$. If $h_k^i(j) = h_j^i(k)$, $\mathcal{P}_j$ broadcasts “YES”, otherwise “NO ”.

[4] we can take G as a subgroup of E/F which is an additive abelian group derived by an elliptic curve E defined in finite field F.

(e) Any player $\mathcal{P}_i$ marks as disqualified if
   - $\mathcal{P}_i$ received at least $t+1$ complaints in Step (c), or
   - there are at least $t+1$ players that broadcast "NO "in Step (d).

(f) Each player then builds the set of non-disqualified players $\mathcal{Q}$.

(g) The distributed secret value s is not explicitly computed by any player, but it equals $s = \sum_{i \in \mathcal{Q}} z_i \bmod q$. Each player $\mathcal{P}_i$ computes $h_i(x) = \sum_{j \in \mathcal{Q}} h_i^j(x) = \sum_{j \in \mathcal{Q}} f_j(x, i)$, and sets his share of the secret as $s_i = \sum_{j \in \mathcal{Q}} h_i^j(0) \bmod q$

3. Revealing $y = sP$

(a) Each player $\mathcal{P}_i$, $i \in \mathcal{Q}$, broadcasts $A_{0k}^{(i)} = a_{0k}^{(i)} P$ for $k = 0, 1, 2, \ldots, t$.

(b) Each player $\mathcal{P}_j$, $j \in \mathcal{Q}$ verifies the values broadcast by the other players in $\mathcal{Q}$, namely, for each $i \in \mathcal{Q}$, $\mathcal{P}_j$ checks if

$$h_j^i(0)P = \sum_{k=0}^{t} j^k A_{0k}^{(i)} \tag{2}$$

If the check fails for an index i, $\mathcal{P}_j$ complains against $\mathcal{P}_i$ by broadcasting the value $h_j^i(0)$ that satisfies Eq.1 but do not satisfies Eq.2.

(c) For players $\mathcal{P}_i$ who receive at least one valid complaint, i.e. value which satisfies Eq.1 not Eq.2., the other players can jointly compute $z_i = f_i(0,0)$, $f_i(0,z)$, and $A_{0k}^{(i)}$ for $k = 0, 1, \ldots, t$, with Lagrange interpolation. For all players in $\mathcal{Q}$, set $y_i = A_{00}^{(i)} = z_i P$. Compute $y = \sum_{i \in \mathcal{Q}} y_i$.

3.2 Security Results of BDKG

The security properties of BDKG are stated in the following theorem.

Theorem 2. *In discrete-log based cryptosystems, protocol BDGK is a secure protocol for the distributed generation of keys , namely, it satisfies the correctness and secrecy requirements of definition 1 with threshold t, for any $t < \frac{n-2}{3}$.*

Proof of Correctness. We first note that all honest players in the protocol compute the same set $\mathcal{Q}$ since the determination of which players are to be disqualified depends on public broadcast information which is known to all (honest) players.

(C1) To each $i \in \mathcal{Q}$, then player $\mathcal{P}_i$ has successfully performed the sharing of z_i among all honest players, and each honest player $\mathcal{P}_j (j \in \mathcal{Q})$ receives its share $s_{ij} = h_j^i(0) = f_i(0, j)$ on z_i. With these shares, we can get a unique polynomial $f_i(0, x)$ which satisfies $f_i(0,0) = z_i$. Thus, for any set R of $t+1$ correct shares,

$$z_i = \sum_{j \in R} \gamma_j s_{ij}$$

where γ_j are appropriate Lagrange interpolation coefficients for the set R. Since each honest party $\mathcal{P}_j$ computes its share s_j of s as $s_j = \sum_{i \in \mathcal{Q}} s_{ij}$, then we have that for the set R:

$$s = \sum_{j \in \mathcal{Q}} z_j = \sum_{j \in \mathcal{Q}} \sum_{k \in R} \gamma_k s_{jk} = \sum_{k \in R} \gamma_k (\sum_{j \in \mathcal{Q}} s_{jk}) = \sum_{k \in R} \gamma_k s_k$$

Since this holds for any set of $t+1$ correct shares then s is uniquely defined.

(C2) The value y is computed (by the honest players) as $y = \sum_{i\in\mathcal{Q}} y_i$, where the values of $y_i (= z_i P)$ are derived from information broadcast in the protocol and thus known to all honest players. We need to show that indeed $y = sP$ where $s = \sum_{i\in\mathcal{Q}} z_i$. We will show that for $i \in \mathcal{Q}$, $y_i = z_i P$, and then $y = \sum_{i\in\mathcal{Q}} y_i = \sum_{i\in\mathcal{Q}} z_i P = (\sum_{i\in\mathcal{Q}} z_i)P = sP$. For players $\mathcal{P}_i(\ i \in \mathcal{Q})$ against whom a valid complaint has been issued in Step 3b, value z_i is publicly reconstructed and y_i set to $z_i P$. Now we need to show that for $\mathcal{P}_i$ $(i \in \mathcal{Q})$, against whom a valid complaint has not been issued, the value y_i is set to $A^{(i)}_{00}$. Values $A^{(i)}_{0k}$, $k = 0, ..., t$ broadcast by player $\mathcal{P}_i$ define a t-degree polynomial $\widehat{h}_i(x)$. Since we assume that no valid complaint was issued against $\mathcal{P}_i$ then the under equation is satisfied for all honest players:

$$h^i_j(0)P = \sum_{k=0}^{t} j^k A^{(i)}_{0k}$$

and thus $\widehat{h}_i(x)$ and $f_i(0, x)$ have at least $t+1$ points in common, so $\widehat{h}_i(x) = f_i(0, x)$, and in particular $y_i = A^{(i)}_{00} = f_i(0, 0)P = z_i P$.

(C3) The secret s is defined as $s = \sum_{i\in\mathcal{Q}} z_i$. Note that as long as there is one value z_i in this sum that is chosen at random and independently from other values in the sum, we are guaranteed to have uniform distribution of s. Also note that the secret s and the components z_i in the sum are already determined at the end of Step 2 of BDKG (since neither the values z_i nor the set $\mathcal{Q}$ change later). Let $\mathcal{P}_i$ be a honest player, $i \in \mathcal{Q}$, and suppose the adversary controls t shares of $f_i(x, z)$. Without loss of generality, we assume he knows $h^i_1(x)$, $h^i_2(x)$, $\ldots$, $h^i_t(x)$,i.e, his view $View_A = \{h^i_1(x), h^i_2(x), \ldots, h^i_t(x)\}$. It is easy to show that, for any value s', we can find $b_{ij} \in Z_q$, where $b_{00} = s'$, $b_{ij} = b_{ji}$, $0 \leq i, j \leq t$ such that if

$$f^{'}_i(x, z) = \sum_{i,j=0}^{t} b_{ij} x^i z^j$$

then, $f^{'}_i(x, 1) = h^i_1(x)$, $f^{'}_i(x, 2) = h^i_2(x)$, $\ldots$, $f^{'}_i(x, t) = h^i_t(x)$. That is to say, $View_A$ does not contain any information of z_i, i.e,

Prob[$\mathcal{P}_i$ has secret $z_i | View_A)$]=Prob[$\mathcal{P}_i$ has secret z_i]=$\frac{1}{q}$ for all $z_i \in Z_q$.

Thus, it is independent of the view of the adversary that each honest $\mathcal{P}_i$ of $\mathcal{Q}$ chooses z_i. Hence the secret x is uniformly distributed.

Proof of Secrecy. To show that the adversary is not able to learn any information about the private key s other than the fact that it is the discrete log of the public key y, We provide a simulator SIM for the BDKG protocol. Formally, a simulator is a probabilistic polynomial-time algorithm that given $y \in G$, such that $y = sP$ for some s, can produce a distribution of messages that is indistinguishable from a normal run of the protocol where the players controlled by the simulator are controlled instead by honest players. This is the familiar technique used to show that zero-knowledge proofs do not reveal any private information.

The input to the simulator is a y that could have been established at the end of a normal run of the protocol. In the description and analysis of the simulator we assume, without loss of generality, that the adversary compromises players $\mathcal{P}_1, \ldots, \mathcal{P}_{t'}$, where $t' \leq t$. We denote the indices of the players controlled by the adversary by $\mathcal{B}= \{1, ..., t'\}$, and the indices of the players controlled by the simulator by $\mathcal{G}= \{t'+1, ..., n\}$.

Consider the following algorithm for the simulator, SIM:

1. Perform Steps 1-2 on behalf of the uncorrupted players $\mathcal{P}_{t'+1}, ..., P_n$ exactly as in protocol BDKG. This includes receiving and processing the information sent privately and publicly from corrupted players to honest ones. At the end of Step 2 the following holds:
   – The set $\mathcal{Q}$ is well-defined. Note that $\mathcal{G} \subseteq \mathcal{Q}$ and that polynomials $f_i(x, y)$ for $i \in \mathcal{G}$ are chosen at random.
   – The adversary's view consists of polynomials $f_i(x, y)$ for $i \in \mathcal{B}$, unique polynomials $h^i_j(x)(=f_i(x, j))$ for $i \in \mathcal{Q}$ and $j \in \mathcal{B}$.
   – SIM knows all polynomials $f_i(x, z)$ for $i \in \mathcal{Q}$ (note that for $i \in \mathcal{Q} \cap \mathcal{B}$ the honest parties, and hence SIM, receive enough consistent shares from the adversary that allow SIM to compute all these parties' polynomials $f_i(x, y)$). SIM also knows all $h^i_j(x)(=f_i(x, j))$.
2. Perform the following calculations:
   – Compute $A^{(i)}_{0k} = a^{(i)}_{0k}P$ for $i \in \mathcal{Q} \backslash \{n\}$, $k = 0, ..., t$
   – Set$A^{(n)*}_{00} = y - (\sum_{i \in \mathcal{Q} \backslash \{n\}} A^{(i)}_{00})$.
   – Assign $h^{n*}_j(x) = h^n_j(x) = f_n(x, j)$ for $j = 1, ..., t$
   – Compute $A^{(n)*}_{0k} = \gamma_{k0} A^{(n)*}_{00} + \sum_{i=1}^{t} \gamma_{ki} h^{n*}_i(0)P$ for $k = 1, ..., t$, where γ_{ki} are the Lagrange interpolation coefficients.
3. Broadcast $A^{(i)}_{0k}$ for $i \in \mathcal{G} \backslash \{n\}$, and $A^{(n)*}_{0k}$ for $k = 0, ..., t$.
4. Perform the checks of step 3 of the algorithm for each player $\mathcal{P}_j$,$j \in \mathcal{G}$, on the $A^{(i)}_{0k}$, $i \in \mathcal{B}$, broadcast by the players controlled by adversary and broadcast any necessary complaints.
5. Perform Step 3c of the protocol on behalf of the uncorrupted parties, i.e. compute $f_i(0, x)$ and z_i in the clear with Lagrange interpolation for every $\mathcal{P}_i$ against whom a valid accusation was broadcast in the previous step.

We will show that the view of the adversary $\mathcal{A}$ that interacts with SIM on input y is the same as the view of $\mathcal{A}$ that interacts with the honest players in a regular run of the protocol that outputs the given y as the public key.

In a regular run of protocol BDKG, $\mathcal{A}$ sees the following probability distribution of data produced by the uncorrupted parties:

– Polynomials $h^i_j(x)(=f_i(x, j))$ for $i \in \mathcal{G}$ and $j \in \mathcal{B}$, randomly chosen in $Z_q[x]$, (since $f_i(x, y)$, which is a bivariate symmetrical polynomial over Z_q of degree t, is randomly chosen).

– Values $A_{0k}^{(i)}$, $i \in \mathcal{G}$, $k = 0, ..., t$ that correspond to exponents of coefficients $a_{0k}^{(i)} (i \in \mathcal{G}, k = 0, ..., t)$ of randomly chosen bivariate symmetrical polynomials $f_i(x, z)$.

Since here we are interested in runs of BDKG that end with the value y as the public key output of the protocol, we note that the above distribution of values is induced by the choice (of the good players) of polynomials $f_i(x, z)$ for $i \in \mathcal{G}$, uniformly distributed in the family of t-degree bivariate symmetrical polynomials over Z_q subject to the condition that

$$\sum_{i \in \mathcal{Q}} A_{00}^{(i)} = y.$$

In other words, t-degree bivariate symmetrical polynomials $f_i(x, z)$ $(i \in \mathcal{G} \backslash \{n\})$ and $f_n(x, z)$ over Z_q which are randomly chosen satisfy

$$f_n(0, 0) = \log_P^y - \textstyle\sum_{i \in \mathcal{Q} \backslash \{n\}} f_i(0, 0) mod\ q,$$

where $\log_P^y$ denotes logarithm s if $y = sP$, P is a generator of group G.

We show that the simulator SIM outputs a probability distribution which is identical to the above distribution. First note that the above distribution depends on the set $\mathcal{Q}$ defined at the end of Step 2 of the protocol. Since all the simulator's actions in Step 1 of the simulator are identical to the actions of honest players interacting with $\mathcal{A}$ in a real run of the protocol, thus we are assured that the set $\mathcal{Q}$ is defined at the end of this simulation step identically to its value in the real protocol. We now describe the output distribution of SIM in terms of t-degree polynomials $f_i^*(x, z)$ corresponding to the choices of the simulator when simulating the actions of the honest players and defined as follows:

– $f_i^*(x, z) = f_i(x, z)$ for $i \in \mathcal{G} \backslash \{n\}$
– $f_n^*(0, 0) = \log_P^{A_{00}^{(n)*}}$, $f_n^*(x, j) = h_j^{n*}(x) = h_j^n(x) = f_n(x, j)$ for $j = 1, \ldots, t$.

It can be seen that by this definition that the values of these polynomials evaluated at the points $j \in \mathcal{B}$ coincide with the values $f_i(x, j)$ which are seen by the corrupted players in Step 2a of the protocol. Also, the coefficients of these polynomials agree with the exponentials $A_{0k}^{(i)}$, $i \in \mathcal{G} \backslash \{n\}$ and $A_{0k}^{(n)*}$ published by the simulator in Step 2 on behalf of the honest parties and corresponding to the players' values in Step 3a of the protocol. Thus, these values pass the verifications of Eq.(1) and (2) as in the real protocol.

It remains to be shown that polynomials $f_i^*(x, z)$ belong to the right distribution. Indeed, for $i \in \mathcal{G} \backslash \{n\}$ this is immediate since they are defined identically to $f_i(x, z)$ which are chosen according to the uniform distribution. For $f_n^*(x, z)$ we see $f_n^*(x, j) = f_n(x, j)$ for $j = 1, ..., t$, get random values $f_n^*(0, j) = f_n^*(0, j)$ for $j = 1, ..., t$, and then compute $f_n^*(0, x)$ by $f_n^*(0, 0) = \log_P^{A_{00}^{(n)*}}$ and Lagrange interpolation formula. We set $f_n^*(x, 0) = f_n^*(0, x)$, and compute $f_n^*(x, z)$ by $f_n^*(x, j)$ for $j = 0, ..., t$ and Lagrange interpolation formula. Since $f_n(x, j) (= f_n^*(x, j))$ for $j = 1, ..., t$ are chosen in Step 1 random and independent polynomials, then so is $f_n^*(x, j)$.

3.3 Efficiency

BDKG consists of two phases. In first phase, BDKG does not require expensive computation. In second phase, each player $\mathcal{P}_j$, $j \in \mathcal{Q}$, participating in the distributed key generation in BDKG, computes $t+1$ values $A_{0k}^{(j)} = a_{0k}^{(j)}P$ for $k = 0, 1, 2, \ldots, t$ and performs $(|\mathcal{Q}|-1)$ verifications of form $h_j^i(0)P = \sum_{k=0}^{t} j^k A_{0k}^{(i)}$ where $i \in \mathcal{Q}\backslash\{j\}$ and $|\mathcal{Q}|$ denotes the number of elements of $\mathcal{Q}$. In other words, each player $\mathcal{P}_j$, $j \in \mathcal{Q}$, computes $|\mathcal{Q}|(t+1)$ exponentiations[5] over the group G. Similarly, if BDKG is performed in Z_p, we also have the result that each player $\mathcal{P}_j$, $j \in \mathcal{Q}$, computes $|\mathcal{Q}|(t+1)$ modular exponentiations over Z_p. While each player $\mathcal{P}_j$, $j \in \mathcal{Q}$, participating in the distributed key generation in [5] and [9], computes at least $n(t+1)$ and $(n+|\mathcal{Q}|)(t+1))$ modular exponentiations over Z_p respectively. The computational cost of the schemes in [13, 14] is more than that of schemes in [5] and [9], because the schemes in [13, 14] are based on homomorphic encryption with proof of fairness. Thus, compared with other schemes, our scheme, BDKG, is very efficient computationally.

4 Application of BDKG

We consider the following case: To perform some specific task, e.g, distributed signature, the private key generator (PKG, for short) of ID-based cryptography, etc, a group, $\mathcal{G}$, is initially formed by n players $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$ that jointly generate a pair of public and private keys $< s, y >$ of $\mathcal{G}$ and among whom $t+1$ players can reconstruct the private key s, where $s \in Z_q^*$, $y = sP$, P is a generator of G which is a cycle additive group of prime order q. To be better competent for its works in some environments, for example, mobile ad hoc networks, $\mathcal{G}$ is often required to be highly dynamic and decentralized: new players may join the group $\mathcal{G}$ at any time, and when they do, they must be provided with shares of private key s, by the other players in $\mathcal{G}$ (since it may be unreasonable to assume that a trusted entity is able to provide a new player with such share). The above case that we describe often occurs in lots of application, e.g, [16, 17, 18, 24, 25].

When a good player wants to join $\mathcal{G}$, if the private key s of $\mathcal{G}$ is based on univariate polynomials, we meet the problem introduced in Section 1.1. If the private key s is generated with our new scheme, BDKG, this problem can be completely avoided. The detail is following.

We assume that n players $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$ jointly a symmetrical bivariate polynomial as follows,

$$f(x,z) = \sum_{i=0,j=0}^{t} x^i z^j, and \;\; f(0,0) = s,$$

and each player $\mathcal{P}_i$, $i \in \{1, 2, 3, \ldots, n\}$, gets a share $h_i(x)(= f(x,i))$ of $f(x,z)$ and a share $s_i(= h_i(0) = f(0,i))$ of s. We also assume that new players have

[5] Exponentiation over the additive cycle group G: Given a generation P of G, $x \in Z_q$ where q is a large prime, compute xP.

a secure channel with each player of $\mathcal{G}$. When a new player $\mathcal{P}_{new}$ joins $\mathcal{G}$, he requests and receives corresponding information from at least $t+1$ players from $\mathcal{G}$. More specifically the protocol must be realized as follows:

1. $\mathcal{P}_{new}$ chooses a set $\mathcal{G}_0$ of at least $t+1$ players from $\mathcal{G}$. Without loss of generality, we assume this set of players is $\mathcal{G}_0$={ $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_{t+1}$}.
2. $\mathcal{P}_{new}$ requests to be accepted as a member of $\mathcal{G}$.
3. Each $\mathcal{P}_i$ of $\mathcal{G}_0$ sends to $\mathcal{P}_{new}$ the piece of information $h_i(new)$.
4. Then, $\mathcal{P}_{new}$ computes his share polynomial h_{new} of $f(x,z)$ with $h_{new}(i) = h_i(new)$ and Lagrange interpolation:

$$h_{new}(x) = \sum_{i=1}^{t+1} \prod_{i \neq j} \frac{x-j}{i-j} h_{new}(i) = \sum_{i=1}^{t+1} \prod_{i \neq j} \frac{x-j}{i-j} h_i(new)$$

Apparently, the above protocol is based on the assumption that each player of $\mathcal{G}$ is honest. However, this kind of assumption is impractical or impossible in many contexts. We must consider the case that some players of $\mathcal{G}$ are corrupted. We assume that there are at most t players corrupted in $\mathcal{G}$ and modify the above protocol as follows.

1. $\mathcal{P}_{new}$ chooses a set $\mathcal{G}_0$ of at least n players from $\mathcal{G}$. Without loss of generality, we assume this set of players is $\mathcal{G}_0$={ $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$}.
2. $\mathcal{P}_{new}$ requests to be accepted as a member of $\mathcal{G}$.
3. Each $\mathcal{P}_i$ of $\mathcal{G}_0$ sends to $\mathcal{P}_{new}$ the piece of information $h_i(new)$.
4. Then, $\mathcal{P}_{new}$ computes as follows:

$$(s_{new}, h_{new}(x)) = \textit{EC-Interpolate}(h_1(new), h_2(new), \ldots, h_n(new))$$

In the above modified protocol, we use notation $(v, f(x))$ =*EC-Interpolate*(v_1, v_2, ..., v_n) for public reconstruction of a value through polynomial interpolation with the use of error-correcting codes. If $\{v_1, v_2, \ldots, v_n\}$ is a set of values such that at least $n-t$ of them lie on some $t-$degree polynomial $f(x)$, and $n > 3t+2$, then v =*EC-Interpolate*(v_1, v_2, ..., v_n)=$f(0)$, The polynomial can be computed using any standard error-correction mechanism, e.g. the Berlekamp-Welch decoder [26].

In the above modified protocol, $\mathcal{P}_{new}$ is convinced that s_{new} is a share of private key s held by him. However, if the private key s of $\mathcal{G}$ is based on univariate polynomials and we also assume that $\mathcal{P}_{new}$ gets s_{new} with Lagrange interpolation, we can ensure that private key s is secure with the shuffling technique presented by J. Kong *et al.*[19], but, we can not seem to convince $\mathcal{P}_{new}$ that s_{new} is a share of private key s held by him.

5 Conclusion

In this paper, we present a distributed key generation based on bivariate symmetric polynomials, BDKG. Compared with others solutions, our construction is efficient in computation and simple and flexible in applications. BDKG is proven secure only against a static adversary. However, we don't know if it is secure or not against an adaptive adversary. In addition, we also simply discuss its application.

References

1. 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.
2. Y. Desmedt. Society and group oriented cryptography: A new concept. In CRYPTO'87, volume 293 of LNCS, pages 120-127. Springer-Verlag, 1987.
3. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In CRYPTO'89, volume 435 of LNCS, pages 307-315. Springer-Verlag, 1990.
4. 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.
5. T.P. Pedersen. A threshold cryptosystem without a trusted party. In Eurocrypt'91, volume 547 of LNCS, pages 522-526. Springer-Verlag, 1991.
6. R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Eurocrypt'99, pages 90-106, 1999. LNCS No. 1592.
7. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Eurocrypt'97, pages 103-118, 1997. LNCS No. 1233.
8. V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Eurocrypt'98, pages 1-16, 1998. LNCS No.1403.
9. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In Eurocrypt'99, pages 295-310. Lecture Notes in Computer Science 1592 (1999).
10. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin. Secure distributed key generation for discrete-Log based cryptosystems, Journal of Cryptology, May 2006, OF1-OF33.
11. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adap tive security for threshold cryptosystems. In Crypto'99, LNCS No. 1666. Springer-Verlag, Berlin, 1999, pp.98-115.
12. John Canny, Stephen Sorkin. Practical Large-scale Distributed Key Generation. In Eurocrypt'04, pages 138-152. Lecture Notes in Computer Science 3027.
13. 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.
14. Rui Zhang, Hideki Imai. Round Optimal Distributed Key Generation of Threshold Cryptosystem based on Discrete Logarithm Problem. In ACNS'03, volume 2846 of LNCS, pages 96-110.
15. S. Jarecki and A. Lysyanskaya. Adaptively Secure Threshold Cryptography : Introducing Concurrency, Removing Erasures. In Eurocrypt'00, LNCS 1807, pages 221-242. Springer-Verlag, 2000.
16. Bo Zhu, Guilin Wang, Zhiguo Wan, Mohan S. Kankanhalli, Feng Bao, Robert H. Deng. Providing Robust Certification Services Against Active Attacks in Ad Hoc networks. Proc. 24th IEEE International Per formance Computing

and Communications Conference (IPCCC 2005), Phoenix, 7-9 April, 2005. http://citeseer.ist.psu.edu/722019.html
17. A. Khalili, J. Katz, and W.A. Arbaugh. Toward secure key distribution in truly Ad-Hoc networks, Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), IEEE Computer Society, 2003, pp. 342-346.
18. H. Deng, A. Mukherjee, D.P. Agrawal. Threshold and identity-based key management and authentication for wireless Ad Hoc networks, International Conference on Information Technology: Coding and Computing (ITCC 804), vol. 1, 2004, pp. 107-115.
19. J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, Providing robust and ubiquitous security support for MANET, IEEE ICNP 2001, 2001.
20. D. Boneh and M. Franklin. Efficient Generation of Shared RSA Keys. In Crypto'97, LNCS 1294, pages 425-439. Springer-Verlag, 1997.
21. Y. Frankel, P. Gemmel, Ph. MacKenzie, and M. Yung. Proactive RSA. In Crypto'97, LNCS 1294, pages 440-454. Springer-Verlag, 1997.
22. G. Poupard and J. Stern. Generation of Shared RSA Keys by Two Parties. In Asiacrypt'98, LNCS 1514, pages 11-24. Springer-Verlag, 1998.
23. D.R.Stinson , R.Wei. Unconditionally Secure Proactive Secret Sharing Scheme with Combinatorial Structures. In: Proc. of the 6th Annual Workshop on Selected Areas in Cryptography (SAC'99). LNCS 1758, Springer Verlag, 1999. 200-214.
24. M. Narasimha, G. Tsudik, and J. H. Yi. On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In IEEE International Conference on Network Protocols, pages 336-345, Nov. 2003.
25. Nitesh Saxena, Gene Tsudik, Jeong Hyun Yi. Efficient Node Admission for Short-lived Mobile Ad Hoc Networks, IEEE International Conference on Network Protocols (ICNP'05), November 2005.
26. E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.
27. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In: ACM Symposium on the Theory of Computing. (1988) 1-10.
28. Naor, M., Pinkas, B., Reingold, O.: Distibuted Pseudo-Random Functions and KDCs. In: EUROCRYPT'99. Volume 1592 of LNCS. (1999) 327-346.
29. Blundo, C., Santis, A.D., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-Secure Key Distribution for Dynamic Conferences. In: CRYPTO'92. Volume 740 of LNCS. (1999) 471-48.

This article was processed using the LaTeX macro package with LLNCS style