Framework of Threshold Signature Scheme

AMIS
November 10, 2022

Contents

1 Distributed Key Generation

Distributed key generation (Abbrev. DKG) is a cryptographic process in which multiple parties participate to the calculation of a shared public and a private key set. The main merit of distributed key generation is not rely necessarily on trusted third parties. Let $\mathscr{G}$ be an elliptic curve group with order $N_{\mathscr{G}}$ and G be a fixed point of $\mathscr{G}$. In our implementation, the threshold t, the number of total participants n, and the level n_i of each participants $\mathscr{P}_i$ are determined before someone performs the following DKG protocol:

Algorithm 1 Distributed Key Generation

Input: positive integers t, n and a non-negative integer n_i.

Output: two sets of positive integers $\{n_i\}_{i=1}^n$ $\{x_i\}_{i=1}^n$, a share s_i, and the unique corresponding public Key P.

1: Each participant $\mathscr{P}_i$

a. randomly chooses x-coordinate $x_i \in [1, N_{\mathscr{G}} - 1]$, and $\{u_{i,j}\}_{j=1}^{t-1} \in [0, N_{\mathscr{G}} - 1]$ (i.e. $f_i(x) := \sum_{j=0}^{t-1} u_{i,j} x^j$).

b. computes $u_{i,0} \cdot G$ and hash commitment $[\mathscr{C}_i, \mathscr{D}_i] = \mathrm{H}(u_{i,0} \cdot G)$. Here H is a crypto-hash function.

c. broadcasts x-coordinate x_i, the level n_i, and the hash commitment $\mathscr{C}_i$.

2: Each participant $\mathscr{P}_i$

a. computes the associated Birkhoff coefficient w_i and Feldman's commitment $F_{i,j} := u_{i,j} \cdot G$.

b. broadcasts own decommitment $\mathscr{D}_i$ and Feldman's commitment $F_{i,j}$.

3: Each participant $\mathscr{P}_i$

a. verifies the decommitment $\mathscr{D}_i$.

b. computes the values $f_i^{(n_k)}(x_k)$ for all $1 \le k \le n$.

c. sends the value $f_i^{(n_k)}(x_k)$ to the participant $\mathscr{P}_k$.

4: Each participant $\mathscr{P}_i$

a. verifies the Feldman commitment for $f_k^{(n_i)}(x_i)$ for all $1 \le k \ne i \le n$.

b. computes the public key $\sum_{i=1}^n u_{i,0} \cdot G$, the own share $s_i := \sum_k f_k^{(n_i)}(x_i)$, the point $s_i \cdot G$ and Schnorr's proof of $s_i \cdot G$.

c. broadcasts $s_i \cdot G$ and Schnorr's proof of $s_i \cdot G$.

5: Each participant $\mathscr{P}_i$

a. verifies the Schnorr's proof of $s_k \cdot G$ for all $1 \le k \ne i \le n$ and the Public key is given by by $\sum_{k=1}^n w_k \cdot (s_k \cdot G) = \sum_{k=1}^n u_{k,0} \cdot G$.

---

2 Threshold Signature

A (t, n)-threshold signature scheme is a digital signature scheme where any t or more signers of a group of n signers can produce signatures.

2.1 Hierarchical threshold signature

This implementation is sign protocols in section 4.2 GG18 or Section 3.3: Signing. Let $\mathscr{G}$ be an elliptic curve group with order $N_{\mathscr{G}}$ and G be a fixed point of $\mathscr{G}$. Let $S \subset [1..n]$ be the set of players participating in the signature protocol. We assume that $|S| \geq t$. We note that using the appropriate Birkhoff coefficients $b_{i,S}$ each player in S can locally map its own (t, n) share s_i of the private key into a $(t, |S|)$ share of x, $w_i = (b_{i,S})(x_i)$.

Algorithm 2 Sign: Part 1

Input: a Hash message m, a threshold t, the share w_i, the public Key P, a set of levels $\{n_i\}_{i=1}^n$ and a set of x-coordinates $\{x_i\}_{i=1}^n$.

Output: (r, s) **or** $\perp$.

1: Each participant $\mathscr{P}_i$

a. generates a private Key and the associated public Key P_i of a given homomorphic scheme.

b. randomly chooses k_i and $a_i \in [0, N_{\mathscr{G}} - 1]$.

c. computes the proof of the public key for P_i.

d. computes hash commitment $[\mathscr{C}_i, \mathscr{D}_i] = \text{Hash}(a_i \cdot G)$.

e. broadcasts the digest $\mathscr{C}_i$, the public Key P_i, and the proof of P_i. In Paillier case, the proof is the integer factorization for P_i. In CL scheme case, the proof is Schnorr's proof.

Algorithm 3 Sign: Part 2

2: Each participant $\mathscr{P}_i$

a. verifies the proof of publicKey P_i.

b. runs sMtA with the inputs $a = k_i$ and $b = a_i$ to get $\alpha_{i,j}$ and $\beta_{i,j}$, and the inputs $a = k_i$ and $b = w_i$ to get $v_{i,j}$ and $\mu_{i,j}$.

c. computes $\delta_i := k_i a_i + \sum_{j,j\neq i} \alpha_{i,j} + \sum_{j,j\neq i} \beta_{j,i} \mod N_{\mathscr{G}}$ and $s_i := k_i w_i + \sum_{j,j\neq i} v_{i,j} + \sum_{j,j\neq i} \mu_{j,i} \mod N_{\mathscr{G}}$.

d. broadcasts δ_i.

3: Each participant $\mathscr{P}_i$

a. computes $\delta := \sum_i \delta_i$ and $\delta^{-1} \mod N_{\mathscr{G}}$ and Schnorr's proof of a_i.

b. broadcasts Schnorr's proof for a_i and the decommitment $\mathscr{D}_i$.

4: Each participant $\mathscr{P}_i$

a. verifies the decommitment $\mathscr{D}_i$ and Schnorr's proof for a_i.

b. computes $R = (R_x, R_y) := \delta^{-1} \cdot (\sum_i a_i \cdot G)$. If R is the identity element, then stop.

c. computes $\tilde{s}_i := \sum_i R_x s_i + m k_i$.

d. randomly chooses ℓ_i, and $\rho_i \in [0, N_{\mathscr{G}} - 1]$.

e. computes $V_i := \tilde{s}_i \cdot R + \ell_i \cdot G$, and $A_i := \rho_i \cdot G$ and hash commitment $[\mathfrak{C}_i, \mathfrak{D}_i] := \text{Hash}(V_i, A_i)$.

f. broadcasts the digest $\mathfrak{C}_i$.

5: Each participant $\mathscr{P}_i$

a. computes Schnorr's proof for ρ_i and Schnorr's proof for $\tilde{s}_i$ and ℓ_i.

b. broadcasts the decommitment $\mathfrak{D}_i$ and the above two Schnorr's proofs.

6: Each participant $\mathscr{P}_i$

a. verifies the decommitment $\mathfrak{D}_i$, and Schnorr's proof for ρ_i and Schnorr's proof for $\tilde{s}_i$ and ℓ_i.

b. computes $V := (-m) \cdot G + (-R_x) \cdot P + \sum_i V_i$, $A := \sum_i A_i$, $U_i := \rho_i \cdot A$, $T_i := \ell_i \cdot A$ and hash commitment $[\mathbb{C}_i, \mathbb{D}_i] := \text{Hash}(U_i, T_i)$.

c. broadcasts $\mathbb{C}_i$.

7: Each participant $\mathscr{P}_i$

a. verifies the decommitment $\mathbb{D}_i$ and $\sum_i U_i = \sum_i T_i$.

b. broadcasts $\tilde{s}_i$.

8: Each participant $\mathscr{P}_i$

a. computes $s := \sum_i \tilde{s}_i$. If $s = 0$, then stop.

b. verifies that the signature (r, s) is correct with input m, and the public key P.

2.2 Multiplication to Addition

In this subsection, we introduce the sMtA protocol: it appears in Section 5: Removing the ZK proofs from the MtA protocol. The goal of the protocol permits that Alice(resp. Bob) has own input a(resp. b) such that they get α and β respectively with $ab = \alpha + \beta$ in a certain field eventually. Meanwhile, Alice(resp. Bob) can not learn any knowledge of b and β (resp. a and α). For achieving this goal, homomorphic encryption schemes plays an important role.

2.2.1 Paillier's MtA

Algorithm 4 Paillier's simplified Multiplication to Addition (abbrev. sMTA).

Input: Alice's secret $a \in [0, N_{\mathscr{G}} - 1]$ **and Bob's secret** $b \in [0, N_{\mathscr{G}} - 1]$**.**
Output: Alice gets α **and Bob gets** β **such that** $\alpha + \beta = a \cdot b \mod N_{\mathscr{G}}$**.**

1: Alice initiates the protocol by

a. sending $c_A = E_A(a)$.

2: Bob computes $c_B = b \times_E c_A +_E E_A(\beta')$, where β' is chosen uniformly at random in $[0, N_{Pai} - (N_{\mathscr{G}} - 1)^2]$. And set $\beta = -\beta' \mod N_{\mathscr{G}}$.

a. send c_B, the Schnorr's proofs of $B = b \cdot G$ and $B' := \beta' \cdot G$ to Alice.

3: Alice

a. verifies Schnorr's proof of B and B'.

b. verifies $\alpha \cdot G = a \cdot B + B'$.

c. decrypts c_B to obtain α' and sets $\alpha = \alpha' \mod N_{\mathscr{G}}$.

2.2.2 MtA of CL Scheme

Algorithm 5 CL Multiplication to Addition (with check).

Input: Alice's secret $a \in [0, N_{\mathscr{G}} - 1]$ **and Bob's secret** $b \in [0, N_{\mathscr{G}} - 1]$**.**
Output: Alice gets α **and Bob gets** β **such that** $\alpha + \beta = a \cdot b \mod N_{\mathscr{G}}$**.**

1: Alice initiates the protocol by

  a. sending $c_A = E_A(a)$.

  b. (with check) computing the associated proof $proof(c_A)$.

2: Bob computes $c_B = b \times_E c_A +_E E_A(\beta')$, where β' is chosen uniformly at random in $[0, N_{\mathscr{G}} - 1]$. And set $\beta = -\beta' \mod N_{\mathscr{G}}$.

  a. (with check) Verify that the $proof(c_A)$ is correct form.

  b. send c_B to Alice.

  c. (with check) send $G_\beta := \beta \cdot G$ and $G_b := b \cdot G$.

3: Alice

  a. decrypts c_B to obtain α' and sets $\alpha = \alpha' \mod N_{\mathscr{G}}$.

  b. (with check) verifies $\alpha \cdot G + G_\beta = a \cdot G_b$.

3 Reshare

Let $\mathscr{G}$ be an elliptic curve group with order $N_{\mathscr{G}}$ and G be a fixed point of $\mathscr{G}$. In our implementation, the threshold t, the number of total participants n, and the level n_i of each participants $\mathscr{P}_i$ have been determined. The following implementation of resharing is the standard algorithm:

Algorithm 6 Reshare

Input: a threshold t, the public Key P, a set of levels $\{n_i\}_{i=1}^n$, a set of x-coordinate $\{x_i\}_{i=1}^n$ and a share s_i.

Output: a new share $\tilde{s}_i$

1: Each participant $\mathscr{P}_i$

a. randomly chooses $u_{i,0} = 0$, and $\{u_{i,j}\}_{j=1}^{t-1} \in [0, N_{\mathscr{G}} - 1]$ (i.e. $f_i(x) := \sum_{j=0}^{t-1} u_{i,j} x^j$).

b. computes Birkhoff coefficient w_i and Feldman's commitment $F_{i,j} := u_{i,j} \cdot G$.

c. broadcasts the Feldman's commitments $F_{i,j}$.

2: Each participant $\mathscr{P}_i$

a. computes the values $f_i^{(n_k)}(x_k)$ for all $1 \le k \le n$.

b. sends the value $f_i^{(n_k)}(x_k)$ to the participant $\mathscr{P}_k$.

3: Each participant $\mathscr{P}_i$

a. verifies the Feldman commitment for $f_k^{(n_i)}(x_i)$ for all $1 \le k \ne i \le n$.

b. computes the new share $\tilde{s}_i := s_i + \sum_k f_k^{(n_i)}(x_i)$, the point $\tilde{s}_i \cdot G$, and Schnorr's proof of $\tilde{s}_i \cdot G$.

c. broadcasts $\tilde{s}_i \cdot G$ and Schnorr's proof of $\tilde{s}_i \cdot G$.

4: Each participant $\mathscr{P}_i$

a. verifies the Schnorr's proof of $\tilde{s}_k \cdot G$ for all $1 \le k \ne i \le n$ and the Public key is given by $P = \sum_{k=1}^n w_k \cdot (\tilde{s}_k \cdot G)$.