TSS-BIP32 Simple Flow

AMIS
June 24, 2021

Contents

Let G be the base point of the elliptic curve group of secp256k1, and secp256k1_N be the order of this group.

Algorithm 1 Master key share

Input: Alice's(i.e. $\mathscr{P}_0$) private seed s_0, and the rank n_0. Bob's(i.e. $\mathscr{P}_1$) private seed s_1 and the rank n_1

Output: a share $\tilde{s}_i$, Birkhoff parameter (x_i, n_i), the associated public key P, and the chain-code C.

1: Each participant $\mathscr{P}_i$ (i.e. means Alice and Bob)

a. Randomly chooses a random value $r_i \in [0, \text{secp256k1}_N]$.

b. Randomly chooses a x-coordinate $x_i \neq 0$.

c. Generates a garbled circuit $\mathscr{G}_i$ with his input: r_i and the seed s_i.

d. Broadcasts the x-coordinate x_i, the rank n_i, and the generating garbled circuit $\mathscr{G}_i$ to $\mathscr{P}_{1-i}$

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

a. Verifies the set (x_i, n_i) can be recovered Birkhoff coefficient.

b. Performs OT to get the other people's random wires.

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

a. Uses obtained random wires and the garbled circuit $\mathscr{G}_i$ to learn the secret called $\hat{s}_i$ (i.e. $= \mathrm{I}_L + r_{1-i}$) and the same chain code C. Here $I := \text{HMAC512}(\text{"Bitcoin seed"}, s_0 || s_1)$ with $I = I_L || I_R$, where the byte length of I_L is equal to 32.

b. Broadcasts hash commitments $H(\hat{s} \cdot G)$ and $H(r_i \cdot G)$.

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

a. Randomly chooses $a_{i,1} \in [0, \text{secp256k1}_N]$.

b. Broadcasts the decommitments of $H(\hat{s}_i \cdot G)$ and $H(r_i \cdot G)$ and $a_{i,1} \cdot G$.

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

a. Verifies the decommitments of $H(\hat{s}_i \cdot G)$ and $H(r_i \cdot G)$. If the verification is failure, then stop it.

b. Verifies $P := \hat{s}_0 \cdot G - r_1 \cdot G = \hat{s}_1 \cdot G - r_0 \cdot G$. If the verification is failure or P is the identity element of the elliptic curve group, then stop it. Let P be the public Key.

c. Sets $f_i(x) := a_{i,1} * x + \hat{s}_i - r_i$ and computes $f_i^{n_i}(x_i) \mod \text{secp256k1}_N$ and $f_i^{n_{1-i}}(x_{1-i}) \mod \text{secp256k1}_N$.

d. Sends $f_i^{n_{1-i}}(x_{1-i})$ to the participant P_{1-i}.

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

a. Verifies the Feldmann commitment of $f_{1-i}^{n_i}(x_i)$. If the verification is failure, then stop it.

b. Sets the share as $\tilde{s}_i := \frac{f_i^{n_i}(x_i) + f_{1-i}^{n_i}(x_i)}{2} \mod \text{secp256k1}_N$.

Algorithm 2 Child key share

Input: Alice's(i.e. $\mathscr{P}_0$) private parent share s_0, the associated public key P, the chain-code C, the key-index i, and her Birkhoff parameter (x_0, n_0) . Bob's(i.e. $\mathscr{P}_1$) private parent share s_1, the associated public key P, the key-index i, the chain-code C, and his Birkhoff parameter (x_1, n_1)

Output: a share $\tilde{s}_i$, the associated public key P_{child}, and the chain-code C_{child}.

1: Each participant $\mathscr{P}_i$(i.e. means Alice and Bob)

a. Generates a garbled circuit $\mathscr{G}_i$ with his input: s_i, the chain-code C and the key-index i.

b. Computes $s_i \cdot G$ and its Schnorr proof.

c. Broadcasts Schnorr proof.

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

a. Verifies Schnorr proof and $b_0 \cdot (s_0 \cdot G) + b_1 \cdot (s_1 \cdot G) = P$. If the verification is failure, then stop it.

b. Performs Quid Pro Quo-tocols: Strengthening Semi-Honest Protocols with Dual Execution.

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

a. Learns the $I = \text{HMAC-SHA512}(\text{Key} = C, \text{Data} = 0x00||\text{ser}_{256}(\text{"private key"})||\text{ser}_{32}(i))$ (ref: the notations can be found in the official document of Bip32).

b. Splits I into two 32-byte sequences, I_L and I_R. Here $I = I_L||I_R$ with the byte length of I_L and I_R are both 32.

c. If $\text{parse}_{256}(I_L) \geq \text{secp256k1}_N$, then stop it.

d. If $P_{child} := \text{parse}_{256}(I_L) \cdot G + P$ is the identity element in the elliptic curve group, then stop it.

e. Sets the chain-code is $C_{child} := I_R$ and the child share is $\tilde{s}_i := s_i + \frac{\text{parse}_{256}(I_L)}{2}$ mod secp256k1_N.