Modify MPC-CMP as a Threshold Signature Scheme

Modified parts are marked in red.

0 Definitions

Definition 1.1(F-VSSS, Feldman's Verifiable Secret Sharing). Define the Verifiable Secret Sharing Scheme as the four tuple $(com, share, recover, vrfy)$. In cryptography, a secret sharing scheme is verifiable if auxiliary information is included that allows players to verify their shares as consistent. More formally, verifiable secret sharing ensures that even if the dealer is malicious there is a well-defined secret that the players can later reconstruct.

Given $ssid = (\ldots, \mathbb{G}, q, g, P)$.

1. Let $(\tilde{F}, \tilde{f}) \leftarrow \mathcal{M}(com, \Pi^{F-VSSS}, t, ssid)$:
   - Sample $f^1, \ldots, f^{t-1} \leftarrow F_q$.
   - For $k \in \{1, \ldots, t-1\}$, set $F^k = g^{f^k}$, $\tilde{F} = (F^k)_k$, $\tilde{f} = (f^k)_k$.
2. Let $(c, \tilde{X}, \tilde{x}) \leftarrow \mathcal{M}(share, \Pi^{F-VSSS}, t, \mathbf{P}, ssid; \tilde{f}, x)$.
   - Define a polynomial:
     $f(\nu) = x + f^1 * \nu + f^2 * \nu^2 + \cdots + f^{t-1} * \nu^{t-1}$
   - Compute $X = g^x$.
   - For $j \in \mathbf{P}$, set $x^j = f(j)$, $X^j = g^{x^j}$, $\tilde{X} = (X^j)_j$, $\tilde{x} = (x^j)_j$.
   - Set Feldman's commitment $c = (X, \tilde{F})$
3. Verify $\mathcal{M}(vrfy, \Pi^{F-VSSS}, j, c, ssid; x_j) = 1$.

   The secret share x_j could be validated as follows:
   - Compter $c_j^0 = X$
   - Compter $c_j^1 = F_1^{(j^1)}$
   - Compter $c_j^2 = F_2^{(j^2)}$
   - $\ldots$
   - Compter $c_j^{t-1} = F_{t-1}^{(j^{t-1})}$
   - Verify $\Pi_k^{t-1} c_j^k = g_j^x$
4. Let $(x) \leftarrow \mathcal{M}(recover, \Pi^{F-VSSS}, S \subseteq \mathbf{P}, (X, \tilde{F}), ssid; \tilde{x})$.

   The origin secret x could be computed with at least t shares via the Sharmir's Secret Sharing Scheme.
   - Verify $|S| \geq t$.
   - Verify the $F - VSSS$ commitment of the shares.
     - For $j \in S$, verify share x_j
       - Compter $c_j^0 = X$
       - Compter $c_j^1 = F_1^{j^1}$
       - Compter $c_j^2 = F_2^{j^2}$
       - $\ldots$
       - Compter $c_j^{t-1} = F_{t-1}^{j^{t-1}}$
       - Verify $\Pi_k^{t-1} c_j^k = g_j^x$
   - For $j \in S$, Compute $\ell_j(0) = \prod_{i \neq j}^{i \in S} \frac{j}{j-i} \pmod q$

- Compute the secret $x = \sum_{j \in S} \ell_j(0) \cdot x^j \pmod{q}$

5. Let $(X) \leftarrow \mathcal{M}(recover\_pub, \Pi^{F-VSSS}, \tilde{X}, S \subseteq \mathbf{P}, ssid)$.

   With at least t public keys corresponding to the secret shares, the public key corresponding to the origin secret could be computed as follows:

   - Verify $|S| \geq t$.
   - For $j \in S$, Compute the Lagrangian interpolation coefficients: $\ell_j(0) = \prod_{i \neq j}^{i \in S} \frac{j}{j-i} \pmod{q}$.
   - Compute the public key $X = \prod_{j \in S} \ell_j(0) \cdot X^j$

1 Protocol: Threshold Key Generation

Round 1.

Upon activation on input $(\mathbf{keygen}, sid, i, t)$ from P_i, interpret $sid = (\ldots, G, q, g, \mathbf{P})$, and do:

- Sample $\{x_i \leftarrow F_q\}$, and set $X_i = g^{x_i}$
- Sample $rid_i \leftarrow \{0,1\}^\kappa$ and compute $(A_i, \tau) \leftarrow \mathcal{M}(com, \Pi^{sch})$.
- Compute $(B_i, r) \leftarrow \mathcal{M}(com, \Pi^{sch})$.
- Compute the shares of x_i.
  - Compute $(c_i, \tilde{f}_i) \leftarrow \mathcal{M}((com, \Pi^{F-VSSS}, t, ssid)$
  - Compute: $((X_i, \tilde{F}_i), \tilde{X}_i, \tilde{x}_i) \leftarrow \mathcal{M}(share, \Pi^{F-VSSS}, t, \mathbf{P}, ssid; \tilde{f}, x)$. Set $X_i^j = g^{x_i^j}$, $\tilde{X}_i = (X_i^j)_j$, $\tilde{x}_i = (x_i^j)_j$
- Sample $u_i \leftarrow \{0,1\}^\kappa$ and set $V_i = \mathcal{H}(sid, i, rid_i, X_i, A_i, B_i, \tilde{X}_i, c_i, u_i))$.

Round 2.

When obtaining (sid, j, V_j) from all P_j, broadcast $(sid, i, rid_i, X_i, A_i, B_i, \tilde{X}_i, c_i, u_i)$ and send (sid, i, x_i^j) to all P_j.

Round 3.

1. Upon receiving $(sid, j, rid_j, X_j, A_j, B_j, u_j, \tilde{X}_j, c_i, x_j^i)$ from P_j, do:
   - Verify $\mathcal{H}(sid, j, rid_j, X_j, A_j, B_j, u_j, \tilde{X}_j, c_i) = V_j$.
   - Verify $\mathcal{M}(vrfy, \Pi^{F-VSSS}, i, c_i, ssid; x_j^i) = 1$.
2. When obtaining the above from all P_j, do:
   - Set $x_i' = \sum_{j \in P} x_j^i$
   - For $j \in P$, set $X_j' = \prod_{k \in P} X_k^j$.
   - Set $\tilde{X}' = (X_j')_j$
   - Set $rid = \oplus_j rid_j$.
   - Compute $\psi_i = \mathcal{M}(prove, \Pi^{sch}, (sid, i, rid), X_i; x_i, \tau)$.
   - Compute $\phi_i = \mathcal{M}(prove, \Pi^{sch}, (sid, i, rid), X_i'; x_i', r)$

   Send (sid, i, ψ_i, ϕ_i) to all P_j.

Output.

1. Upon receiving (sid, j, ψ_j) from P_j, interpret $\psi_j = (\hat{A}_j, \ldots)$ and $\phi_j = (\hat{B}_j, \ldots)$, and do:
   - Verify $\hat{A}_j = A_j$.
   - Verify $\mathcal{M}(vrfy, \Pi^{sch}, (sid, j, rid), X_j, \psi_j) = 1$.
   - Verify $\hat{B}_j = B_j$.
   - Verify $\mathcal{M}(vrfy, \Pi^{sch}, (sid, j, rid), X_j^*, \phi_j) = 1$
2. When passing above verification from all P_j, do:
   - Set $X = \Pi_j X_j$.
   - Compute $X' \leftarrow \mathcal{M}(recover\_pub, \Pi^{F-VSS}, ssid, \tilde{X}')$.
   - Verify $X = X'$
   - Set $X_j = X_j'$ for all $j \in P$
   - Set $x_i = x_i'$
   - Output X.

Error. When failing a a verification report the culprit and halt.

Stored State. Store the following: $rid, t, X, \tilde{X} = (X_1, \ldots, X_n), \mathbf{P}$ and x_i.

2 Protocol: Auxiliary Info. & Threshold Key Refresh

Round 1.

On input $(aux - info, ssid, i)$ from P_i, do:

- Sample two $4\kappa - bit$ long safe primes (p_i, q_i). Set $N_i = p_i q_i$.
- Sample $y_i \leftarrow \mathbb{F}_q$ and set $Y_i = g^{y_i}$. Sample $(B_i, \tau) \leftarrow \mathcal{M}(com, \Pi^{sch})$.
- Compute the shares of 0:
  - Compute $(c_i, \tilde{f}_i) \leftarrow \mathcal{M}((com, \Pi^{F-VSSS}, t, ssid)$
  - $(c, \tilde{X}_i, \tilde{x}_i) \leftarrow \mathcal{M}(share, \Pi^{F-VSSS}, t, \mathbf{P}, ssid; \tilde{f}_i, 0)$
  - $X_i^j = g^{x_i^j}$, $\tilde{X}_i = (X_i^j)_j$, $\tilde{x}_i = (x_i^j)_j$.
- Sample $r \leftarrow \mathbb{Z}_{N_i}^*$, $\lambda \leftarrow \mathbb{Z}_{\phi(N_i)}$, set $t_i = r^2 \pmod{N_i}$, and $s_i = t_i^\lambda \pmod{N})_i$.

  Compute $\hat{\psi}_i = \mathcal{M}(prove, \Pi^{prm}, (ssid, i), (N_i, s_i, t_i); \lambda)$
- Sample $(A_i^j, \tau_j) \leftarrow \mathcal{M}(Com, \Pi^{sch})$, for $j \in \mathbf{P}$. Set $A_i = (A_i^j)_j$.
- Sample $\rho_i, u_i \leftarrow 0, 1^\kappa$ and compute $V_i = \mathcal{H}(ssid, i, X_i, \tilde{X}_i, \tilde{F}_i, A_i, B_i, N_i, s_i, t_i, \hat{\psi}_i, \rho_i, u_i)$.

Broadcast $(ssid, i, V_i)$.

Round 2.

When obtaining (sid, j, V_j) from all P_j, broadcast $(ssid, i, X_i, \tilde{X}_i, \tilde{F}_i, A_i, B_i, N_i, s_i, t_i, \hat{\psi}_i, \rho_i, u_i)$ to all.

Round 3.

1. Upon receiving $(ssid, i, X_i, \tilde{X}_i, \tilde{F}_i, A_i, B_i, N_i, s_i, t_i, \hat{\psi}_i, \rho_i, u_i)$ from P_j, do:
   - Verify $N_i \geq 2^{8\kappa}$ and $\mathcal{M}(verify, \Pi^{prm}, (ssid, j), (N_j, s_j, t_j), \hat{\psi}_j) = 1$.
   - Verify $\mathcal{H}(sid, j, X_j, \tilde{X}_i, \tilde{F}_i, A_j, Y_j, B_j, N_j, s_j, t_j, \hat{\psi}_j, \rho_j, u_j) = V_j$.
   - Verify $\mathcal{M}(recover\_pub, \Pi^{F-VSSS}, (id_G, \tilde{F}_j), \mathbf{P}, ssid, \tilde{X}_j) = id_G$.
2. When obtaining the above from all P_j, do:
   - Compute $\psi_i = \mathcal{M}(prove, \Pi^{mod}, (sid, \rho_, i), N_i; (p_i, q_i))$.
   - Compute $\phi_i = \mathcal{M}(prove, \Pi^{fac}, (sid, \rho_, i), (N_i, \kappa); (p_i, q_i))$.
   - For $j \in P$, set $C_i^j = enc_j(x_i^j)$ and $\psi_i^j = \mathcal{M}(prove, \Pi^{sch}, (ssid, \rho, i), X_i^j; x_i^j, \tau_j)$.
   - Compute $\pi_i = \mathcal{M}(prove, \Pi^{sch}, (sid, i, rid), Y_i; y_i, \tau)$.

Send $(sid, i, \psi_i, \phi_i, \pi_i, C_i^j, \psi_i^j)$ to all P_j.

Output.

1. Upon receiving $(sid, j, \psi_j, \phi_j, \pi_j, C_j^i, \psi_j^i)$ from P_j, set $x_j^i = dec_i(C_j^i) \pmod q$ and do:
   - Verify $g^{x_j^i} = X_j^i$. If $g^{x_j^i} \neq X_j^i$ calculate $\mu = (C_j^i \cdot (1+N_i)^{-x_j^i})^{1/N} \pmod N)^2$ and do:

     Dec Error: Send to all (P_j, C_j^i, x_j^i, μ) to each P_j.
   - Verify $\mathcal{M}(vrfy, \Pi^{F-VSSS}, j, (id_G, \tilde{F}_j), ssid; x_j^i) = 1$.
   - Verify $\mathcal{M}(prove, \Pi^{mod}, (sid, \rho_, j), N_j; (p_j, q_j)) = 1$ and $\mathcal{M}(prove, \Pi^{fac}, (sid, \rho_, j), (N_j, \kappa); (p_j, q_j)) = 1$
   - Interpret $\pi_j = (\hat{B}_j, \ldots)$, and Verify $\hat{B}_j = B_j$ and $\mathcal{M}(prove, \Pi^{sch}, (sid, j, rid), Y_j; y_j, \tau) = 1$.
   - For $k \in P$, interpret $\psi_j^k = (\hat{A}_j^k, \ldots)$, and verify $\hat{A}_j = A_j$ and verify $\mathcal{M}(vrfy, \Pi^{sch}, (sid, rho, i), X_j^k, \psi_j^k) = 1$.
2. When passing above verification from all P_j, do:
   - Set $x_i = x_i + \sum_j x_j^i \pmod q$.
   - Set $X_j = X_j + \Pi_{k \in P} X_k^j$ for $j \in P$ and $\tilde{X} = (X_k)_k$
   - Compute $X' \leftarrow \mathcal{M}(recover\_pub, \Pi^{F-VSS}, ssid, \tilde{X}^*)$.
   - Verify $X' = X$
   - Output $(ssid, i, \tilde{X} = (X_j)_j, \tilde{Y} = (Y_j)_j, \tilde{N} = (N_j)_j, \tilde{s} = (s_j)_j, \tilde{t} = (t_j)_j)$.

Error. When failing a a verification or receiving a Dec Error report the culprit and halt.

Stored State. Store the following: (x_i, y_i, p_i, q_i).

3 Protocol: ECDSA Pre-Signing and Signing

The key shard needs to preprocessed.

On input $(key-shard-preprocess, ssid, i, S \subseteq \mathbf{P}, \tilde{X} = (X_j)_j, \tilde{Y} = (Y_j)_j, \tilde{N} = (N_j)_j, \tilde{s} = (s_j)_j, \tilde{t} = (t_j)_j, (x_i^*, y_i, p_i, q_i))$:

- Compute the Lagrangian interpolation coefficients:

$$\ell_i(0) = \prod_{i \neq j}^{j \in S} \frac{i}{i-j} \pmod q.$$

- Set $x_i = \ell_i(0) \cdot x_i \pmod q$.
- Set $X_j = \ell_i(0) \cdot X_j$, for every $j \in \mathbf{G}$.
- Output $(ssid, i, S \subseteq \mathbf{P}, \tilde{X} = (X_j)_j, \tilde{Y} = (Y_j)_j, \tilde{N} = (N_j)_j, \tilde{s} = (s_j)_j, \tilde{t} = (t_j)_j)$, and (x_i, y_i, p_i, q_i).

You can invoke the ECDSA Pre-Signing and Signing Protocol with the preprocessed key.

4 Some typos In MPC-CMP

In the ZK Proof Π^{mul*} (FIGURE 31, P68) :

- The sampling of r_y in step 1is redandant and r_y is never used.

  $r_y \leftarrow Z^*_{N_1}$
- The computation of A in step 1 should be:

  $A = C^\alpha * r^{N_0} \pmod N)^2_0$