Silence Laboratories

1 Dynamic Secret Sharing

Lagrange coefficients. Let $P(\cdot)$ be a polynomial of degree t and let T be a set of $t+1$ points $(x_i, y_i)_{i\in T}$ then for every x we have $P(x) = \sum_{i\in T} y_i \cdot \ell(x)$, where $\ell_i(x) = \prod_{j\in T, j\neq i} \frac{j-x}{j-i}$.

For a set of $t+1$ points, T, we define $\lambda_{i,T} = \ell_i(0) = \prod_{j\in T, j\neq i} \frac{j}{j-i}$. Then, $P(0) = \sum_{i\in T} y_i \cdot \lambda_{i,T}$.

Problem statement. There are two sets of parties, the old group are the P_i's and new group are the P'_i's, as follows.

- Old group: There are n parties: P_i, $i \in \{1, \ldots, n\}$. The parties have a Shamir sharing of a secret x, namely, each P_i holds s_i such that there exists a degree-t polynomial P with $P(i) = s_i$ for all i, and $P(0) = x$.
- The group element $X = x \cdot G$ is public, as well as all X_i's, where $X_i = s_i \cdot G$. Let T be set of $t+1$ points, then $X = \sum \lambda_{i,T} \cdot X_i = (\sum \lambda_{i,T} \cdot s_i) \cdot G = x \cdot G$.
- New group: There are m parties P_i, $i \in \{1, \ldots, m\}$ who wish to obtain shares s'_i such that there exists a degree-t' polynomial Q with $Q(i) = s_i$ for all i, and $Q(0) = x$.

Generic solution with semi-honest parties

1. Choose a committee of $t+1$ parties from the old group. Without loss of generality, let them be $P_1, \ldots, P_{t+1}$.
2. Let λ_i^x, $i \in \{1, \ldots, t+1\}$, be the Lagrange coefficients for computing $P(x)$, that is, $P(x) = \sum_{i=1}^{t+1} \lambda_i^x \cdot s_i$.
3. Each party P_i, $i \in \{1, \ldots, t+1\}$, picks a random degree t' polynomial Q_i, such that $Q_i(0) = s_i$, and sends ${s'}_i^j = Q_i(j)$ to party P'_j, $j \in \{1, \ldots, m\}$.
4. Note that

$$\begin{aligned} x = P(0) = \sum_{i=1}^{t+1} \lambda_i^0 \cdot s_i = \sum_{i=1}^{t+1} \lambda_i^0 \cdot Q_i(0) &= \sum_{i=1}^{t+1} \lambda_i^0 \cdot \left(\sum_{j\in M} \lambda_{j,M}^0 \cdot {s'}_i^j \right) \\ &= \sum_{i=1}^{t+1} \sum_{j\in M} \lambda_i^0 \cdot \lambda_{j,M}^0 \cdot {s'}_i^j \\ &= \sum_{j\in M} \lambda_{j,M}^0 \cdot \sum_{i=1}^{t+1} \lambda_i^0 \cdot {s'}_i^j \end{aligned}$$

   where $M \subset \{1, \ldots, m\}$ is some set of size $t'+1$, and $\lambda_{j,M}^0$ is the Lagrange coefficient associated with P_j when evaluating $Q_i(0)$ using the points of parties in M.

Thus, since $s_i = Q_i(0)$ is linearly shared among $P'_1, \ldots, P'_m$, each party P'_j, $j \in \{1, \ldots, m\}$, computes its final share

$$s'_j = \sum_{i=1}^{t+1} \lambda_i^0 \cdot {s'}_i^j$$

The new shares s'_j are correct since for every subset $M \subset \{1, \ldots, m\}$ of size $t' + 1$, the below equation holds:

$$\sum_{j \in M} \lambda_{j,M}^0 \cdot {s'}_j \quad = \quad \sum_{j \in M} \lambda_{j,M}^0 \cdot \left(\sum_{i=1}^{t+1} \lambda_i^0 \cdot {s'}_i^j \right)$$

Extending to malicious parties. This require each party $P_1, \ldots, P_{t+1}$ to generate Q_i as above (such that $Q_i(0) = s_i$) and secret share s_i in a publicly verifiable manner. See here for PVSS: eprint.iacr.org/2004/201.pdf.

(2/3) to (3/5) Parties Threshold Modification

1. Let the parties be. $\{P_1, P_2, P_3\}$. Let x coordinates of $\{P_1, P_2, P_3\}$ be x_1, x_2, x_3
2. Let the modified quorum for (3/5) be $\{P_1, P_2, P_3, P_4, P_5\}$ parties. Let x coordinates of $\{P_1, P_2, P_3, P_4, P_5\}$ be x_1, x_2, x_3, x_4, x_5. Here n=5 and t'=3
3. Choose a committee of 2 parties from the old group. Let them be $\Delta = \{P_1, P_2\}$.
4. Let x coordinates of P_1 and P_2 be x_1 and x_2 respectively
5. Each player P_1 AND P_2 does the following:

   Selects a random polynomial $g_1(x)$ and $g_2(x)$ respectively of degree at most 2 (t' -1) such that $g_1(0)$=$f(x_1)$ $g_2(0) = f(x_2)$

   [i] P_1 generates shares on $g_1(x)$ for $P_1 : g_{1,1} = g_1(x_1)$

   [ii] P_1 generates shares on $g_1(x)$ for $P_2 : g_{1,2} = g_1(x_2)$ and communicates $g_{1,2}$ to P_2. P_1 generates shares on $g_1(x)$ for $P_3 : g_{1,3} = g_1(x_3)$ and communicates $g_{1,3}$ to P_3 and so on it generates $g_{1,4}$ for P_4 and $g_{1,5}$ for P_5

   [iii] P_2 generates shares on $g_2(x)$ for $P_2 : g_{2,2} = g_2(x_2)$

   [iv] P_2 generates shares on $g_2(x)$ for $P_1 : g_{1,2} = g_1(x_1)$ and communicates $g_{2,1}$ to P_1. P_3 generates shares on $g_2(x)$ for $P_3 : g_{2,3} = g_1(x_3)$ and communicates $g_{2,3}$ to P_3 and so on it generates $g_{2,4}$ for P_4 and $g_{2,5}$ for P_5
6. Each player P_1 , P_2 does the following:

   [i] Generates public constants γ_1^Δ and γ_2^Δ for P_1 and P_2 respectively:

$$\gamma_1^\Delta = \frac{x_2}{x_2 - x_1}$$

$$\gamma_2^\Delta = \frac{x_1}{x_1 - x_2}$$

7. Each player P_1 , P_2 and P_3 does the following:

   [i] Erases their old shares

   [ii] P_1 computes his new shares

$$\Phi_1 = \gamma_1^{\Delta} \times g_{1,1} + \gamma_2^{\Delta} \times g_{2,1}$$

   [iii] P_2 computes his shares:

$$\Phi_2 = \gamma_1^{\Delta} \times g_{2,2} + \gamma_2^{\Delta} \times g_{1,2}$$

   [iv] P_3 computes his new share

$$\Phi_3 = \gamma_1^{\Delta} \times g_{1,3} + \gamma_2^{\Delta} \times g_{2,3}$$

   [iv] P_4 computes his new share

$$\Phi_3 = \gamma_1^{\Delta} \times g_{1,4} + \gamma_2^{\Delta} \times g_{2,4}$$

   [v] P_5 computes his new share

$$\Phi_3 = \gamma_1^{\Delta} \times g_{1,5} + \gamma_2^{\Delta} \times g_{2,5}$$

(2/3) Parties Secret Recovery

1. The set Δ' contains at least t' members . P_1 $P_2$and P_3 recover the secret using Lagrange interpolation method

$$secret = (\gamma_1^{\Delta'} \times \Phi_1) + (\gamma_2^{\Delta'} \times \Phi_2) + (\gamma_3^{\Delta'} \times \Phi_3)$$

Extending to malicious parties. This require each party $P_1, \ldots, P_{t+1}$ to generate Q_i as above (such that $Q_i(0) = s_i$) and secret share s_i in a publicly verifiable manner. See here for PVSS: eprint.iacr.org/2004/201.pdf.

2 Weighted DKG

Input: Each of P_1,..., P_n has a PKI of signing keys $\{pk_1, \ldots, pk_n\}$ and encryption keys $\{ek_1, \ldots, ek_n\}$, its own signing key sk_i and decryption key dk_i, and list of weights $\{w_1, \ldots, w_n\}$. For simplicity, we assume that: P_1 receives the value of the polynomial at the points $\overrightarrow{x^1} = (1, \ldots, w_1)$; P_2 receives the value of the polynomial at the points $\overrightarrow{x^2} = (w_1 + 1, \ldots, w_1 + w_2)$; P_3 receives the value of the polynomial at the points $\overrightarrow{x^3} = (w_1 + w_2 + 1, \ldots, w_1 + w_2 + w_3)$, and so on. Set S as a set with n parties, $S = \{1, \ldots, n\}$. q is the order of the curve. C is a coordinator.

The protocol:

1. Transmission 1 - C to all: C sends a request to generate a key to all parties.
2. Message 1 - all to C: Each party P_i works as follows:
   (a) P_i chooses a random $sid_i \leftarrow \{0,1\}^\kappa$
   (b) For $k \in \{0,\ldots,t-1\}$, P_i chooses a random $u_i^k \leftarrow \mathbb{Z}_q$ and sets $F_i^k = u_i^k \cdot G$. Let $\overrightarrow{F_i} = F_i^0,\ldots,F_i^{t-1}$, $\overrightarrow{u_i} = (u_i^0,\ldots,u_i^{t-1})$, $u_i(x) = \sum_{k=0}^{t-1} u_i^k \cdot x^k$, and $F_i(x) = u_i(x) \cdot G$.
   (c) P_i chooses a random $r_i \leftarrow \{0,1\}^\kappa$ and sets $c_i = H(sid_i||pid_i||weignts||\overrightarrow{F_i}||r_i)$
   (d) P_i sends $(\sigma_i^1, sid_i, weights, c_i)$ to the coordinator C, where $\sigma_i^1 = sign_{sk_i}(1, sid_i, weights, c_i)$.
3. Transmission 2 - C to all: C receives all $(\sigma_i^1, sid_i, weights, c_i)$ messages, and sends $\{(\sigma_i^1, sid_i, weights, c_i)\}_{i\in S}$ to P_i for all $i \in S$.
4. Message 2 - all to C: Each party P_i works as follows:
   (a) P_i verifies that it received $(\sigma_i^1, sid_i, weights, c_i)$ for n parties that it included in the list of participants, that the sid_i that it choose is in list, that $weights$ the same for all parties, that c_i as it sent in the first message appears in the set, and that all signatures are valid. If not, it aborts. If yes, it sets sid to be a collision-resistant hash of S and all $\{sid_j\}_{j\in S}$.
   (b) P_i computes $\pi_i \leftarrow ZKDL_P^t(sid, pid_i, \overrightarrow{F_i}, \overrightarrow{u_i})$ (where $ZKDL^t$ denotes a batch Fiat-Shamir proof of knowledge of the discrete log of t values, and i is the know identity or public-key of P_i).
   (c) P_i sends $(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)$ to the coordinator C, where $\sigma_i^2 = sign_{sk_i}(2, sid, \{c_i\}_{i\in S}, \overrightarrow{F_i}, r_i, \pi_i)$.
5. Transmission 3 - C to all: C receives all $(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)$ messages, and sends $\{(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)\}_{i\in S}$ to P_i for all $i \in S$.
6. Message 3 - all to C: Each party P_i works as follows:
   (a) After receiving all $\{(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)\}_{i\in S}$, P_i verifies that all signatures are valid and are computed on the same sid that it computed.
   (b) For every $j \in S$ $(j \neq i)$:
      (i) P_i verifies that $H(sid_j||pid_j||\overrightarrow{F_j}||r_j) = c_j$ and that all values are valid (i.e., has the correct sid_j and pid_j and overall structure).
      (ii) P_i verifies $ZKDL_V^t(sid, pid_j, \overrightarrow{F_j}, \pi_j) = 1$.
      (iii) If the commitment is not valid or any F_j^k is not a valid point in the curve subgroup of is equal to the identity point, or if the zero-knowledge verification fails, then P_i aborts. Else, it proceeds.
   (c) P_i sets the VSS sharing polynomial to be $F(x) = \sum_{i\in S} F_j(x)$. That is, the kth coefficient F_k of $F(x)$ is set to $\sum_{i\in S} F_j^k$. Denote $\overrightarrow{F} = (F_0,\ldots,F_{t-1})$.
   (d) P_i sets the output public key to be $public\_key = F_0$.
   (e) For every $j \in \{1,\ldots,n\}$, and for every $x \in \overrightarrow{x^j}$, party P_i sets P_j's shares in $F_i(x)$ to be $d_{i\rightarrow j}^x = u_i(x)$ and encrypts $d_{i\rightarrow j}^x$ under P_j's public key ek_j. Denote the ciphertext by $c_{i\rightarrow j}^x$. Denote the set of all these ciphertexts by $\overrightarrow{c_i}$.
   (f) P_i signs on $(sid, \overrightarrow{F}, \overrightarrow{c_i})$. Denote the signature by σ_i^3.
   (g) P_i sends $(\sigma_i^3, sid, \overrightarrow{F}, \overrightarrow{c_i})$ to the coordinator C.

7. Transmission 4 - C to all: C receives all $(\sigma_i^3, sid, \overrightarrow{F}, \overrightarrow{c_i})$ messages, and sends $\{(\sigma_i^3, sid, \overrightarrow{F}, \overrightarrow{c_i})\}_{i\in S}$ to P_i for all $i \in S$.

8. Output: Each party P_i works as follows:

   (a) After receiving all $\{(\sigma_i^3, sid, \overrightarrow{F}, \overrightarrow{c_i})\}_{i\in S}$, P_i verifies that all signatures are valid and are computed on the same *sid* that it computed, and that all parties sent the same $\overrightarrow{F}$.

   (b) For every $x \in \overrightarrow{x^i}$: P_i decrypts all $\{c^x_{j\rightarrow i}\}_{j\in S}$ and sets $d_i^x = \sum_{j\in S} d^x_{j\rightarrow i} \pmod{q}$.

   (c) P_i verifies that $(d^x_{j\rightarrow i}) \cdot G = F_j(x)$ for all $j \in S$, $x \in \overrightarrow{x^i}$ and that $(d_i^x) \cdot G = F(x)$.

   (d) If any check fails, P_i aborts and raises a security alert. Else, P_i outputs the *public_key* and its polynomial shares d_i^x, for $x \in \overrightarrow{x^i}$, which correspond to the weight w_i.

3 Dynamic refresh for Weighted TSS

Input: There are two groups of parties, the old group are the P_i's and new group are the P_i''s, as follows:

- Old group: There are n parties: P_i, $i \in \{1, \dots, n\}$. The parties have a Shamir sharing of a secret x, namely, each P_i holds s_i such that there exists a degree-t polynomial P with $P(i) = s_i$ for all i, and $P(0) = x$. Each P_i, $i \in \{1, \dots, n\}$ has a PKI of signing keys $\{pk_1, \dots, pk_n\}$ of old parties and encryption keys $\{ek\prime_1, \dots, ek\prime_m\}$ of new parties, its own signing key sk_i and decryption key dk_i, and list of weights $\{w_1, \dots, w_n\}$.
- New group: There are m parties: P_i', $i \in \{1, \dots, m\}$ who wish to obtain shares s_i' such that there exists a degree-t' polynomial Q with $Q(i) = s_i$ for all i, and $Q(0) = x$. Each $P_i\prime$, $i \in \{1, \dots, m\}$ has a PKI of signing keys $\{pk_1, \dots, pk_n\}$ and encryption keys $\{ek_1, \dots, ek_n\}$ of old parties, its decryption key dk_i', and list of new $weights' = \{w_1', \dots, w_m'\}$.

For simplicity, we assume that: P_1 has the value of the polynomial at the points $\overrightarrow{x^1} = (1, \dots, w_1)$; P_2 has the value of the polynomial at the points $\overrightarrow{x^2} = (w_1 + 1, \dots, w_1 + w_2)$; P_3 has the value of the polynomial at the points $\overrightarrow{x^3} = (w_1 + w_2 + 1, \dots, w_1 + w_2 + w_3)$, and so on. The same for new parties P_i', $i \in \{1, \dots, m\}$ with new weights $\{w_1', \dots, w_m'\}$.

C is a coordinatior. q is the order of the curve.

Set S as a set with n_1 parties, which will participate in the dynamic refresh protocol. S' as a set with m new parties, $S' = \{1, \dots, m\}$.

The protocol: First, the parties from the set S calculate their additive shares $\hat{s}_i$ using Lagrange interpolation.

1. Transmission 1 - C to all: C sends a request to run dynamic refresh protocol to all parties in S.
2. Message 1 - all to C: Each party P_i from the set S works as follows:
   (a) P_i chooses a random $sid_i \leftarrow \{0,1\}^\kappa$
   (b) Set $u_i^0 = \hat{s_i}$, and $F_i^0 = u_i^0 \cdot G$
   (c) For $k \in \{1, \dots, t-1\}$, P_i chooses a random $u_i^k \leftarrow \mathbb{Z}_q$ and sets $F_i^k = u_i^k \cdot G$. Let $\overrightarrow{F_i} = F_i^0, \dots, F_i^{t-1}$, $\overrightarrow{u_i} = (u_i^0, \dots, u_i^{t-1})$, $u_i(x) = \sum_{k=0}^{t-1} u_i^k \cdot x^k$, and $F_i(x) = u_i(x) \cdot G$.
   (d) P_i chooses a random $r_i \leftarrow \{0,1\}^\kappa$ and sets $c_i = H(sid_i||pid_i||weights'||\overrightarrow{F_i}||r_i)$
   (e) P_i sends $(\sigma_i^1, sid_i, weights', c_i)$ to the coordinator C, where $\sigma_i^1 = sign_{sk_i}(1, sid_i, weights', c_i)$.
3. Transmission 2 - C to all: C receives all $(\sigma_i^1, sid_i, weights', c_i)$ messages,
   and sends $\{(\sigma_i^1, sid_i, weights', c_i)\}_{i \in S}$ to P_i for all $i \in S$.
4. Message 2 - all to C: Each party P_i from the set S works as follows:
   (a) P_i verifies that it received $(\sigma_i^1, sid_i, weights', c_i)$ for n_1 parties that it included in the list of participants, that the sid_i that it choose is in list, that $weights'$ the same for all parties, that c_i as it sent in the first message appears in the set, and that all signatures are valid. If not, it aborts. If yes, it sets sid to be a collision-resistant hash of S and all $\{sid_j\}_{j \in S}$.
   (b) P_i computes $\pi_i \leftarrow ZKDL_P^t(sid, pid_i, \overrightarrow{F_i}, \overrightarrow{u_i})$ (where $ZKDL^t$ denotes a batch Fiat-Shamir proof of knowledge of the discrete log of t values, and i is the know identity or public-key of P_i).
   (c) P_i sends $(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)$ to the coordinator C, where $\sigma_i^2 = sign_{sk_i}(2, sid, \{c_i\}_{i \in S}, \overrightarrow{F_i}, r_i, \pi_i)$.
5. Transmission 3 - C to all: C receives all $(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)$ messages, and sends $\{(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)\}_{i \in S}$ to P_i for all $i \in S$.

6. Message 3 - all to C: Each party P_i works as follows:

   (a) After receiving all $\{(\sigma_i^2, sid, \overrightarrow{F_i}, r_i, \pi_i)\}_{i\in S}$, P_i verifies that all signatures are valid and are computed on the same sid that it computed.

   (b) For every $j \in S$ $(j \neq i)$:

      (i) P_i verifies that $H(sid_j||pid_j||\overrightarrow{F_j}||r_j) = c_j$ and that all values are valid (i.e., has the correct sid_j and pid_j and overall structure).

      (ii) P_i verifies $ZKDL_V^t(sid, pid_j, \overrightarrow{F_j}, \pi_j) = 1$.

      (iii) If the commitment is not valid or any F_j^k is not a valid point in the curve subgroup of is equal to the identity point, or if the zero-knowledge verification fails, then P_i aborts. Else, it proceeds.

   (c) P_i sets the VSS sharing polynomial to be $F(x) = \sum_{i\in S} F_j(x)$. That is, the kth coefficient F_k of $F(x)$ is set to $\sum_{i\in S} F_j^k$. Denote $\overrightarrow{F} = (F_0, \ldots, F_{t-1})$.

   (d) For every $j \in \{1, \ldots, n\}$, and for every $x \in \overrightarrow{x^j}$, party P_i sets P_j''s shares in $F_i(x)$ to be $d^x_{i\longrightarrow j} = u_i(x)$ and encrypts $d^x_{i\longrightarrow j}$ under P_j's public key ek_j. Denote the ciphertext by $c^x_{i\longrightarrow j}$. Denote the set of all these ciphertexts by $\overrightarrow{c_i}$.

   (e) P_i signs on $(sid, \overrightarrow{F_i}, \pi_i, \overrightarrow{F}, \overrightarrow{c_i})$. Denote the signature by σ_i^3.

   (f) P_i sends $(\sigma_i^3, sid, \overrightarrow{F_i}, \pi_i, \overrightarrow{F}, \overrightarrow{c_i})$ to the coordinator C.

7. Transmission 4 - C to all: C receives all $(\sigma_i^3, sid, \overrightarrow{F_i}, \pi_i, \overrightarrow{F}, \overrightarrow{c_i})$ messages, and sends $\{(\sigma_i^3, sid, \overrightarrow{F_i}, \pi_i, \overrightarrow{F}, \overrightarrow{c_i})\}_{i\in S}$ to P_i' for all $i \in S'$.

8. Output: Each party P_i' works as follows:

   (a) After receiving all $\{(\sigma_i^3, sid, \overrightarrow{F_i}, \pi_i, \overrightarrow{F}, \overrightarrow{c_i})\}_{i\in S}$, P_i verifies that all signatures are valid and are computed on the same sid that it computed, and that all old parties sent the same $\overrightarrow{F}$.

   (b) P_i' verifies that $F(x) = \sum_{i\in S} F_j(x)$.

   (c) P_i' verifies that expected public_key is equal to the F_0.

   (d) For every $j \in S$:

      (i) P_i' verifies $ZKDL_V^t(sid, pid_j, \overrightarrow{F_j}, \pi_j) = 1$.

      (ii) If any F_j^k is not a valid point in the curve subgroup of is equal to the identity point, or if the zero-knowledge verification fails, then P_i' aborts. Else, it proceeds.

   (e) For every $x \in \overrightarrow{x^i}$: P_i' decrypts all $\{c^x_{j\longrightarrow i}\}_{j\in S}$ and sets $d_i^x = \sum_{j\in S} d^x_{j\longrightarrow i} \pmod{q}$.

   (f) P_i' verifies that $(d^x_{j\longrightarrow i}) \cdot G = F_j(x)$ for all $j \in S$, $x \in \overrightarrow{x^i}$ and that $(d_i^x) \cdot G = F(x)$.

   (g) If any check fails, P_i' aborts and raises a security alert. Else, P_i' outputs its polynomial shares d_i^x, for $x \in \overrightarrow{x^i}$, which correspond to the weight w_i'.

References

[1] Yehuda Lindell, Simple Three-Round Multiparty Schnorr Signing with Full Simulatability, protocol 6.1 https://eprint.iacr.org/2022/374.pdf