ECDSA Key Refresh

Parameters

1. x_i^{old}: the old private share key, generated by keygen algorithm
2. n: numbers of parties participant in refresh
3. t: threshold to refresh
4. T: set of parties

Protocol

Step 1. Each party P_i generates a public/private key pair for promise protocol, includes ElGamal key pairs (pk_i', sk_i') and CL key pairs $(\hat{pk}_i, \hat{sk}_i)$.

Step 2. $P_i \in T$ selects a polynomial $p_i(X) = x_i^{old} + \sum_{k=1}^{t} a_{i,k} X^k$ mod p with constant term x_i^{old} and $a_{i,k}$ is generated at random, computes and broadcast $V_{i,k} = a_{i,k} G_{k \in [t]}$. Computes $\sigma_{i,j} = p_i(j)$ for each $j \in [n]$ and sends to P_j respectively.

Step 3. Each party P_i received $\sigma_{j,i}$ from parties in T, verify if $\sigma_{j,i} G = X_j^{old} + \sum_{k \in [t]} V_{j,k}^{ik}$ and computes $x_i^{new} = \sum_{j \in [t]} \lambda_j \sigma_{j,i}$, where X_j^{old} is public share key of each party and $\lambda_j = \prod_{k \in [t], k \neq i} \frac{-k}{j-k}$ is Lagrange basis polynomials. Computes $X_i^{new} = x_i^{new} G$ and generate a NIZK proof $ZKPoK_{X_i^{new}} \{(x_i^{new}) : X_i^{new} = x_i^{new} G\} \to \pi_i$, broadcast$X_i^{new}$ and π_i.

Step 4. Each party P_i verify received π_j from all other parties.

Output

1. public signing key: same as the old public signing key
2. CL secret key: $\hat{sk}_i$
3. ELGamal secret key: sk_i'
4. private share key: x_i^{new}
5. set of public share keys: $[X_i^{new}]_{i \in [n]}$