GG19

1 Original Protocol

We follow the construction of Gennaro and Goldfeder from the eprint version of the paper [1]. We describe the protocol from the point of view of party P_i. We implicitly assume that all messages arrive (otherwise abort).

PROTOCOL 1.1 (GG19 Distributed Key Generation)

The protocol runs between n parties: $P_1, ..., P_n$. the parties run on input threshold t and elliptic curve parameters.

1. **Commitment Round:** Broadcast a commit to random point $Y_i = u_i \cdot G$
2. Broadcast decommitment to Y_i. Check correctness for $n-1$ received decommitments. Otherwise *abort*
3. **VSS round:** Perform (t, n) Feldman-VSS of the value u_i. set the group public key to be $Y = \sum_j Y_j$. set the local secret share to be $x_i = \sum_j f_j(i)$.
4. Broadcast zkPoK of x_i. Verify $n-1$ zkPoK of DLog, otherwise *abort*
5. **Paillier keygen:** Generate Paillier keypair and broadcast the public key e_i
6. Broadcast zkPoK of p_i, q_i such that $N_i = p_i q_i$ (N_i being Paillier modulus associated with e_i). Verify $n-1$ proofs, otherwise *abort*

PROTOCOL 1.2 (GG19 Distributed Signing)

The protocol runs between t parties: $P_1, ..., P_t$. All parties know m, the message to be signed.

1. Compute new t-additive secret share $w_i = x_i\lambda_i$ where λ_i is Lagrangian coefficient.
2. Compute new local public keys (for all j: $W_j = X_j^{\lambda_j}$)
3. **Commitment Round:** Broadcast a commit to random point $\gamma_i \cdot G$
4. **MtA:** Choose random k_i. for all $j \neq i$ do:
   (a) Send $c_i = E_{e_i}(k_i)$ to P_j
   (b) generate and send zk range proof,proving $k_i < K$ where K is chosen such that $N_i > K^2 q$. Verify $t-1$ range proofs, otherwise *abort*
   (c) compute and send $c_{ji} = \gamma_i \times_{e_j} c_j +_{e_j} E_{e_j}(\beta'_j)$ where β'_j is chosen at random from Z_{N_j}. Set $\beta_{ji} = -\beta'_j$
   (d) generate and send a zk range proof that c_{ji} decrypts to a value $< K$ (in the paper it says $b < K$. need to double check)
   (e) set $\alpha_{ij} = D_{d_i}(c_{ij})$
5. **MtAwc:** for all $j \neq i$ do:
   (a) Send $c_i = E_{e_i}(k_i)$ to P_j.
   (b) generate and send zk range proof, proving $k_i < K$ where K is chosen such that $N_i > K^2 q$, Verify $t-1$ range proofs, otherwise *abort*
   (c) compute and send $c_{ji} = w_i \times_{e_j} c_j +_{e_j} E_{e_j}(\nu'_j)$ where ν'_j is chosen at random from Z_{N_j}. Set $\nu_{ji} = -\nu'_j$
   (d) generate and send a zk range proof that c_{ji} decrypts to a value $< K$ (in the paper it says $b < K$. need to double check). Verify $t-1$ range proofs, otherwise *abort*
   (e) Generate and send zkPoK with witness $\{w_i, \nu_{ji}\}$ such that $W_i = w_i \cdot G$ and $c_{ji} = w_i \times_{e_j} c_j +_{e_j} E_{e_j}(\nu'_j)$. Verify $t-1$ proofs, otherwise *abort*
   (f) set $\mu_{ij} = D_{d_i}(c_{ij})$
6. Broadcast $\delta_i = k_i\gamma_i + \sum_{j\neq i} \alpha_{ij} + \sum_{j\neq i} \beta_{ji}$. Set $\delta = \sum_j \delta_j$.
7. Decommit to $\gamma_i \cdot G$. check correctness for $t-1$ received decommitments. otherwise *abort*
8. Generate and Broadcast zkPoK of DLog for γ_i. Verify $t-1$ zkPoK, otherwise *abort*
9. Compute $R = \delta^{-1} \sum_j \Gamma_j$ where $\Gamma_j = \gamma_j \cdot G$. Compute $r = H'(R)$ where H' is hash from group to scalar. Set $s_i = mk_i + r\sigma_i$, where $\sigma_i = k_i w_i + \sum_{j\neq i} \mu_{ij} + \sum_{j\neq i} \nu_{ji}$
10. **Commitment Round:** Compute $V_i = s_i \cdot R + l_i \cdot G$ and $A_i = \rho_i \cdot G$ where l_i, ρ_i are chosen at random. Broadcast a commitment to $\{V_i, A_i\}$
11. Broadcast Decommitment to $\{V_i, A_i\}$. Check correctness for $t-1$ received decommitments (add explicitly Abort when comm-decomm is not correct), otherwise *abort*
12. Generate and broadcast zkPoK with witness $\{s_i, l_i\}$ to prove $V_i = s_i \cdot R + l_i \cdot G$. Verify $t-1$ zkPoK, otherwise *abort*
13. Generate and broadcast zkPoK with witness ρ_i to prove $A_i = \rho_i \cdot G$ (typo in paper). Verify $t-1$ zkPoK, otherwise *abort*
14. **Commitment Round:** Compute $V = -m \cdot G - r \cdot Y + \sum_j V_j$ and $A = \sum_j A_j$. Broadcast commitment to $\{U_i, T_i\} = \{V^{\rho_i}, A^{l_i}\}$
15. Decommit to $\{U_i, T_i\}$. Check correctness of $t-1$ decommitments and check $\sum_j U_j = \sum_j T_j$. Otherwise *abort*
16. Broadcast s_i. ECDSA verify the signature ($s = \sum_j s_j, r$). If True output (s, r). Otherwise *abort*

2 KZen Version

KZen version [2] follows the original protocol with the following changes:

- In step (10) we add $B_i = l_i \cdot A_i$ to the commitment. We use B_i in step (12) to prove with the same witness the statement $\{V_i = s_i \cdot R + l_i \cdot G, B_i = l_i \cdot A_i\}$
- In MtAwc, we replace the proof in (e) with the following:

1. zkPoK of DLog for $\nu'_j \cdot G$
2. check that $k_i \cdot W_j + \nu'_j \cdot G = \mu_i \cdot G$

- We change step (4) from MtA to be MtAwc. To do so we need P_j to know Γ_i. In the original protocol this information is decommited at step (7). In our protoocl we keep the decommitment correctness check but reveal Γ_i at MtAwc step (c). This is secure as long as all commitments from step (3) arrived. We make sure at step (7) that the decomitted Γ_i is the same one used in the MtAwc. Finally we use the same alternative proof as in previous bullet.
- We do not use range proofs in MtA and MtAwc (steps (b) and (d)). See [1] section 5 for reasoning

References

[1] https://eprint.iacr.org/2019/114. First version

[2] https://github.com/KZen-networks/multi-party-ecdsa. Commit b68db7a