Fouque-Stern One Round Distributed Key Rotation Notes

Akilesh Tangella, Drew Stone

akilesh@webb.tools, drew@webb.tools
Github: @saiakilesh, @drewstone

Abstract

We discuss ZenGo's one round Fouque-Stern distributed key rotation (FS-DKR) of threshold ECDSA key shares. All errors are my own fault.

1 Notation

$[n] = \{1, 2, ..., n\}$.

2 Cryptographic Primitive: The Paillier Cryptosystem

The Paillier cryptosystem is an encryption/decryption scheme with some nice properties about which we can generate zero-knowledge proofs. We will see what types of proofs we need in the sections below. We will not discuss the mathematics behind the cryptosystem and just take it as given.

3 ECDSA

In ECDSA, we have a group G (that originates from some fancy elliptic curve math) that consists of elliptic curve points and an element $g \in G$ of prime order q. We choose an ECDSA secret key s by choosing an integer uniformly at random from the set $[q-1]$. The corresponding public key is the elliptic curve point g^s.

4 Fouque-Stern Distributed Key Generation (FS-DKG)

Let G, g, q be defined as in the section above. Assume we have $n \geq 2t + 1$ parties, of which at most t are malicious (in other words $\geq t + 1$ are honest). The output of FS-DKG is for each party $i \in [n]$:

1. An elliptic curve group element tpk that corresponds to a group public key.
2. A secret key share $tsk_i \in [q-1]$.

In the remainder of this section, we describe how FS-DKG generates these outputs in a single round of publicly broadcasted communication. We assume each party $i \in [n]$ has a public/secret keypair for the Paillier cryptosytem, pk_i and sk_i. The communication proceeds as follows. Each party $i \in [n]$ does the following:

1. Chooses an integer s_i randomly from $[q-1]$.
2. Picks a degree t polynomial f_i with coefficients $a_{i,0}, ..., a_{i,t}$, such that $a_{i,0} = f_i(0) = s_i$.
3. Publicly broadcasts each of the following. For each $j \in [n]$:
   (a) The group element $g^{f_i(j)}$.
   (b) The group elements $g^{s_i}, g^{a_{i,1}}, ..., g^{a_{i,t}}$.
   (c) The Paillier encryption $ENCRYPT(f_i(j))$ of $f_i(j)$ under the public key pk_j.
   (d) A proof that steps 1 and 3 are consistent. That is, the discrete logarithm of the group element from step 1 is the value encrypted in step 3.

Each party i then validates:

1. The proof from step 3(d) above for each party $j \in [n]$.
2. That for each party j, $g^{f_j(i)} = g^{s_j} (g^{a_{j,1}})^i ... (g^{a_{j,t}})^{i^t}$ (this is very similar to Feldman's VSS scheme).

Let Q be the set of parties j for which the above validation passes. Note $|Q| \geq t+1$ since there are at least $t+1$ honest parties. Then for each $j \in Q$, $tsk_j = \sum_{i \in Q} f_i(j)$ (can be found using $DECRYPT$, the Paillier decryption scheme under secret key sk_j). And $tpk = \prod_{i \in Q} g^{f_i(0)}$. Note tpk can be computed by any party since step 2(a) is publicly broadcasted.

One instructive way to look at things occurs when we let $f = \sum_{i \in Q} f_i$. Then tsk_j for $j \in Q$ is simply $f(j)$. And the group public key tpk is $g^{f(0)}$ (with corresponding ECDSA secret key $f(0)$). Since f is of degree at most t and $|Q| \geq t+1$, the honest parties Q can come together to recover f and in turn $f(0)$ and generate an ECDSA signature using the secret key. Of course, this method of generating a signature is unsophisticated since it requires reconstructing the secret key in a single location.

5 Converting Shamir Shares into Additive Shares

In FS-DKG, each party $i \in Q$ received $tsk_i = f(i)$ and the secret $s = f(0)$. Let the first $t+1$ of these parties in Q form the set R and let their indices be $i_1, ..., i_{t+1}$, in ascending order. Let V be the $t+1 \times t+1$ square matrix whose (m, n) $(1 \leq m, n \leq t+1)$ entry is i_m^{n-1}. Let a be the vector of coefficients of f: $(a_0, ..., a_t)$. Let w be the vector $(f(i_1) ..., f(i_{t+1}))$. Then the following equality holds:

$$Va = w$$

V is known as a Vandermonde matrix, and it is well-known that V is invertible. So:

$$a = V^{-1} w$$

Let the first row of V^{-1} be $c_{i_1}, ..., c_{i_{t+1}}$. Then:

$$s = f(0) = a_0 = c_{i_1} f(i_1) + ... + c_{i_{t+1}} f(i_{t+1})$$

We will use the fact that s can be expressed as a linear combination of $f(i_1), ..., f(i_{t+1})$ in the next section.

6 Fouque-Stern Distributed Key Rotation (FS-DKR)

Let R be the set of size $t+1$ as in the section above. Let $c_{i_1}, ..., c_{i_{t+1}}$ be as in the section above, as well. Each party $i \in R$ has a tsk_i, tpk, and a pair of Paillier keys pk_i and sk_i. This is because it has already run FS-DKG. The single round of communication for FS-DKR works as follows. Each party $i \in R$ does the following:

1. Picks a degree t polynomial h_i with coefficients $b_{i,0}, ..., b_{i,t}$, such that $b_{i,0} = h_i(0) = c_i tsk_i$.
2. Publicly broadcasts each of the following. For each $j \in R$:
    (a) The group element $g^{h_i(j)}$.
    (b) The group elements $g^{c_i tsk_i}, g^{b_{i,1}}, ..., g^{b_{i,t}}$.
    (c) The Paillier encryption $ENCRYPT(h_i(j))$ of $h_i(j)$ under the public key pk_j.
    (d) A proof that steps 1 and 3 are consistent. That is, the discrete logarithm of the group element from step 1 is the value encrypted in step 3.

Each party i then validates:

1. The proof from step 2(d) above for each party $j \in R$.
2. That for each party $j \in R$, $g^{h_j(i)} = g^{tsk_j} \left(g^{b_{j,1}}\right)^i \dots \left(g^{b_{j,t}}\right)^{i^t}$ (this is very similar to Feldman's VSS scheme).

For each $j \in R$, the new tsk_j is $\sum_{i \in Q'} h_i(j)$ (can be found using $DECRYPT$, the Paillier decryption scheme under secret key sk_j). And the new tpk stays the same.
Since $\sum_{j \in R} h_j(0) = \sum_{j \in R} c_j tsk_j = s$. We have that the new tsk_j's are a valid secret key sharing of the group public key tpk.

The downfall of this approach is that it assumes that honest parties from FS-DKG remain honest in FS-DKR. Whether this is a valid assumption is a good question.

Each party then generates new Paillier keys, for use in the next round of FS-DKR, along with proofs they were generated correctly. How this is done is beyond the scope of this writeup. But see ZenGo's ZK-PAILLIER library.

7 Adding New Participants

We were not quite right when we said in FS-DKR that each party has already ran FS-DKG. What if we want to add new parties that didn't participate in the initial FS-DKG. This is not so difficult as long as the new party has a pair of Paillier keys (the public key needs to be broadcast to all parties). All we have to do is assign an index i to the new party and then broadcast all the messages from steps 2(a) through 2(d) above to this party. They can then construct a valid secret key share.

8 References

1. https://medium.com/applied-mpc/a-crash-course-on-mpc-part-3-c3f302153929
2. https://github.com/ZenGo-X/fs-dkr/