Threshold EdDSA Submissions of FROST and (maybe) Sparkle

Chelsea Komlo
University of Waterloo
Zcash Foundation, Dfns

Thanks for everyone in the FROST submission team for discussion, and Tim Ruffing.

Sept. 26, 2023
TLDR

- We will be submitting a FROST submission!
- Is there practical interest in a Sparkle submission?
**Multi-sigs**

- **MuSig** [MPSW'18, BDN'18]  
  Assumptions: DL+ROM  
  Signing Rounds: 3

- **SimpleMuSig** [BDN'18, CKM'21]

- **MuSig2** [NRS'21]
  Assumptions: AOMDL+ROM  
  Signing Rounds: 2

- **DWMS** [AB'21]

- **SpeedyMuSig** [CKM'21]

**Threshold**

- **Lindell22**

- **Sparkle** [CKM'23]

- **FROST** [KG'20, BCKMTZ'22]

- **FROST2** [CKM'21]

- **FROST3** [RRJSS'22, CJRS'23]
  Assumptions: AOMDL+ROM  
  Signing Rounds: 2

**Algebraic One-More-Discrete Log (AOMDL):**
- stronger assumption
- partially non-interactive schemes

- All are concurrently secure ✔
FROST

Key Generation:

\((sk_i, PK_i), PK\)

Round 1:
Output \(R_i \leftarrow g^{r_i}, S_i \leftarrow g^{s_i}\)

Round 2:
\(a_i \leftarrow H'(PK, m, i, \{j, R_i, S_i\}_{j \in S})\)
\(R = \prod_{i \in S} R_i S_i^{a_i}\)
\(c \leftarrow H(PK, m, R)\)
Output \(z_i \leftarrow r_i + a_i s_i + c s k_i \lambda_i\)

Combine / Verify:

\(z = \sum_{i \in S} z_i\)
\(\text{sig} = (R, z)\)
\(c \leftarrow H(PK, m, R)\)
\(R \cdot PK^c = g^z\)
FROST Signing

- Two rounds; first round can be preprocessed
- Static security: AOMDL (falsifiable) + ROM
- Adaptive Security: Coming soon!
- Active security; honest minority
- Can be performed over a public channel assuming an untrusted coordinator
FROST Takes an Opinionated Stance

- Simplicity and performance matters
- If the protocol fails, misbehaving parties can be identified and re-run
- Robustness can be implemented at a higher layer (ROAST)
Choice of plain FROST is to not exclude use cases [BCKMTZ22]; multi-scalar multiplication closes the computational gap.

---

**Speeding up FROST with multi-scalar multiplication**

June 1, 2023

by Deirdre Connolly, Conrado Gouvea

We optimized our implementation of FROST by upwards of 50% over the trivial implementation, without changing the protocol and therefore maintaining its existing security guarantees. We use a known trick to do so: multi-scalar multiplication, which is exactly designed to give this kind of performance speedup.

In the FROST threshold signing protocol, we perform many elliptic curve operations for key generation, signing, and signature verification. Because FROST is a Schnorr threshold signing scheme, the signature that is produced is compatible with single-party Schnorr signature verification. As such, there is no additional computation overhead to verifying signatures produced by FROST vs single-party.

However, when performing FROST signing, signers must perform a linear number of group element multiplications, proportionate to the number of signers, as shown below (see the FROST specification for details).

```python
def compute_group_commitment(commitment_list, binding_factor_list):
    # Implementation...
```
FROST Submission Team

- Deirdre Connolly, SandboxAQ
- Elizabeth Crites, Web3 Foundation
- Conrado Gouvea, Zcash Foundation
- Jack Grigg, Electric Coin Company
- Jonathan Katz, University of Maryland & Dfns
- Chelsea Komlo, University of Waterloo & Dfns & Zcash Foundation
- Mary Maller, Ethereum Foundation & PQShield
- Nikita Sorokovikov, Dfns
- Denis Varlakov, Dfns
FROST in Practice, Today
Sparkle

Key Generation:
\((sk_i, PK_i), PK\)

Round 1:
\(R_i \leftarrow g^{r_i}; cm_i = H(m, S, R_i)\)
Output \(cm_i\)

Round 2:
Output \(R_i\)
\(R = \Pi_{i \in S} R_i\)

Round 3:
\(c \leftarrow H(PK, m, R)\)
Output \(z_i \leftarrow r_i + cs k_i \lambda_i\)

Combine / Verify:
\(z = \sum_{i \in S} z_i\)
\(sig = (R, z)\)
\(c \leftarrow H(PK, m, R)\)
\(R \cdot PK^c = g^z \checkmark\)
Sparkle

- Three online rounds;
- Addresses the theoretical question of standard assumptions without expensive ZKPs
- Static security: DL + ROM
- Adaptive Security: AOMDL + AGM + ROM, without erasures
- Active security; honest minority
- Can be performed over a public channel assuming an untrusted coordinator
(My) Opinions

- Protocol flexibility is a good theoretical idea but is a huge source of bugs in practice- downgrade attacks, etc.

- We should aim to design submissions with as few of “moving parts” or choices as possible.

- Don’t push complex and theoretical questions to users and implementors.
Takeaways

• We have a great team working on a FROST submission!

• Is a Sparkle of draft interest to implementors?

• Keeping things simple with as few of choices as possible leads to success and security for implementations.

• We have practical questions, like:
  
  • What ciphersuites should submissions cover?
  
  • Should implementations be fully self-contained (vendors dependencies)?

Thank you!