FROST: Flexible Round-Optimized Schnorr Threshold Signatures

CFRG Update for Threshold Signature Working Group, IETF 110

Chelsea Komlo[1,2] Ian Goldberg[1]

[1] University of Waterloo [2] Zcash Foundation

March 2021

FROST: Summary

- Two-round Schnorr threshold signing protocol, or single-round with preprocessing.
- FROST specifically trades off robustness for round efficiency.
- Signing operations are secure when performed concurrently, against an adversary that controls up to $t - 1$ signers.
- Key generation be performed by either a trusted dealer or via a Distributed Key Generation (DKG) Protocol

FROST: Summary

- Two-round Schnorr threshold signing protocol, or single-round with preprocessing.
- FROST specifically trades off robustness for round efficiency.
- Signing operations are secure when performed concurrently, against an adversary that controls up to $t - 1$ signers.
- Key generation be performed by either a trusted dealer or via a Distributed Key Generation (DKG) Protocol

FROST: Summary

- Two-round Schnorr threshold signing protocol, or single-round with preprocessing.
- FROST specifically trades off robustness for round efficiency.
- Signing operations are secure when performed concurrently, against an adversary that controls up to $t - 1$ signers.
- Key generation be performed by either a trusted dealer or via a Distributed Key Generation (DKG) Protocol

FROST: Summary

- Two-round Schnorr threshold signing protocol, or single-round with preprocessing.
- FROST specifically trades off robustness for round efficiency.
- Signing operations are secure when performed concurrently, against an adversary that controls up to $t - 1$ signers.
- Key generation be performed by either a trusted dealer or via a Distributed Key Generation (DKG) Protocol

Current Status

- FROST was adopted as a working group item at the end of January.
- We are working on the first draft, focusing on implementation details not specified in our paper.
- We are writing a second proof of security using standard assumptions.
- Several parallel implementations exist, but these need to converge.

Current Status

- FROST was adopted as a working group item at the end of January.
- We are working on the first draft, focusing on implementation details not specified in our paper.
- We are writing a second proof of security using standard assumptions.
- Several parallel implementations exist, but these need to converge.

Current Status

- FROST was adopted as a working group item at the end of January.
- We are working on the first draft, focusing on implementation details not specified in our paper.
- We are writing a second proof of security using standard assumptions.
- Several parallel implementations exist, but these need to converge.

Current Status

- FROST was adopted as a working group item at the end of January.
- We are working on the first draft, focusing on implementation details not specified in our paper.
- We are writing a second proof of security using standard assumptions.
- Several parallel implementations exist, but these need to converge.

Feedback from Call for Adoption (summary)

- Compatibility with EdDSA *verification* over Ed25519/Ed448.
  - Using EdDSA-style deterministic nonces is insecure in a multi-signer setting.
  - Our draft will specify signatures compatible with verification specified in RFC 8032.
- Preprocessing may be difficult to perform securely.
- Option for either trusted dealer or DKG.

Feedback from Call for Adoption (summary)

- Compatibility with EdDSA *verification* over Ed25519/Ed448.
  - Using EdDSA-style deterministic nonces is insecure in a multi-signer setting.
  - Our draft will specify signatures compatible with verification specified in RFC 8032.
- Preprocessing may be difficult to perform securely.
- Option for either trusted dealer or DKG.

Feedback from Call for Adoption (summary)

- Compatibility with EdDSA *verification* over Ed25519/Ed448.
  - Using EdDSA-style deterministic nonces is insecure in a multi-signer setting.
  - Our draft will specify signatures compatible with verification specified in RFC 8032.
- Preprocessing may be difficult to perform securely.
- Option for either trusted dealer or DKG.

Feedback from Call for Adoption (summary)

- Compatibility with EdDSA *verification* over Ed25519/Ed448.
  - Using EdDSA-style deterministic nonces is insecure in a multi-signer setting.
  - Our draft will specify signatures compatible with verification specified in RFC 8032.
- Preprocessing may be difficult to perform securely.
- Option for either trusted dealer or DKG.

Feedback from Call for Adoption (summary)

- Compatibility with EdDSA *verification* over Ed25519/Ed448.
  - Using EdDSA-style deterministic nonces is insecure in a multi-signer setting.
  - Our draft will specify signatures compatible with verification specified in RFC 8032.
- Preprocessing may be difficult to perform securely.
- Option for either trusted dealer or DKG.

Next Steps

- Full specification for v1.
- Interoperable implementations.
- Specify prime-order curves, then adapt to curves with cofactors, considering:
  - Point validation during signing.
  - Publishing verification-compatible signatures (RFC 8032).

Next Steps

- Full specification for v1.
- Interoperable implementations.
- Specify prime-order curves, then adapt to curves with cofactors, considering:
  - Point validation during signing.
  - Publishing verification-compatible signatures (RFC 8032).

Next Steps

- Full specification for v1.
- Interoperable implementations.
- Specify prime-order curves, then adapt to curves with cofactors, considering:
  - Point validation during signing.
  - Publishing verification-compatible signatures (RFC 8032).

Next Steps

- Full specification for v1.
- Interoperable implementations.
- Specify prime-order curves, then adapt to curves with cofactors, considering:
  - Point validation during signing.
  - Publishing verification-compatible signatures (RFC 8032).

Next Steps

- Full specification for v1.
- Interoperable implementations.
- Specify prime-order curves, then adapt to curves with cofactors, considering:
  - Point validation during signing.
  - Publishing verification-compatible signatures (RFC 8032).

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.

- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Roadmap

- Within the core draft:
  - Trusted dealer
  - Two round
  - Non-signing signature aggregator.
- Possible future extensions:
  - Preprocessing (single-round signing)
  - DKG (possibly more generally useful)
  - Share recovery/adding new participants

Questions?

Extras: Protocol Specifics

FROST Sign: Round One

Participant i | **Signature Aggregator**

$(d_i, e_i) \xleftarrow{\$} \mathbb{Z}_q^* \times \mathbb{Z}_q^*$

$(D_i, E_i) = (g^{d_i}, g^{e_i})$

$(D_i, E_i) \longrightarrow$

FROST Sign: Round One

Participant i | **Signature Aggregator**

$(d_i, e_i) \xleftarrow{\$} \mathbb{Z}_q^* \times \mathbb{Z}_q^*$

$(D_i, E_i) = (g^{d_i}, g^{e_i})$

$(D_i, E_i) \longrightarrow$

FROST Sign: Round One

Participant i | **Signature Aggregator**

$(d_i, e_i) \xleftarrow{\$} \mathbb{Z}_q^* \times \mathbb{Z}_q^*$

$(D_i, E_i) = (g^{d_i}, g^{e_i})$

$$\xrightarrow{(D_i, E_i)}$$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

$\sigma = (R, z = \sum z_i)$

"binding value" to bind signing shares to ℓ, m, and B

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B) ←

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

→ z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

This step cannot be inverted by anyone who does not know (d_i, e_i).

z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B) ←

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

→ z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B) ←

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

→ z_i

Publish $\sigma = (R, z = \sum z_i)$

FROST Sign: Round Two

Signer i

Signature Aggregator

$B = ((1, D_1, E_1), \ldots, (t, D_t, E_t))$

(m, B)

$\rho_\ell = H_1(\ell, m, B), \ell \in S$

$R = \prod_{\ell \in S} D_\ell \cdot (E_\ell)^{\rho_\ell}$

$c = H_2(R, Y, m)$

$z_i = d_i + (e_i \cdot \rho_i) + \lambda_i \cdot s_i \cdot c$

Signature format and verification are identical to single-party Schnorr.

z_i

Publish $\sigma = (R, z = \sum z_i)$