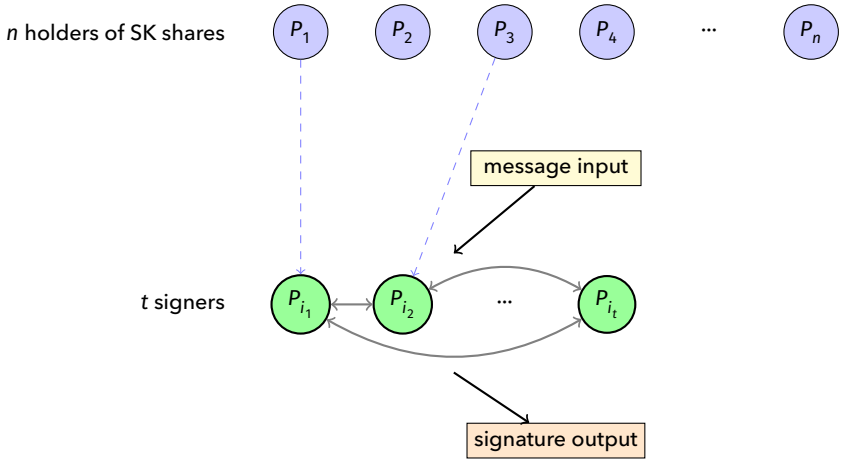
Twig: Two-Round Threshold ECDSA from Non-Interactive Multiplication

Speaker: Yingjie Lyu[1]

[1]Shandong University (recently graduated)

Presented on July 7, 2026 at TCPT2:
NIST Event for Threshold Call Preview Talks #2

Threshold signature: basic idea



In this talk

Overview

Problem statement

Technique 1: Non-Interactive Multiplication

Technique 2: Presignature re-randomization

Implementation and Performance

The paper

Threshold ECDSA in Two Rounds. Yingjie Lyu, Zengpeng Li, Hong-Sheng Zhou, and Xudong Deng. In ACM CCS 2025

Main contributions:

1. Signing only takes 2 rounds
   - Prior state-of-the-art DKLs24 scheme[1] takes 3
   - Concurrently achieved by Dahari-Garbian, Nof, and Parker[2]
2. Fast online computation (<1 ms), constant message size (1.2 – 1.9 KiB)
   - First scheme to enjoy both
   - Total computation (200 ms for 3 parties) also competitive
3. Fixes a security downgrade identified by Groth and Shoup[3]
   - Too many unused presignatures cause issues for prior schemes

[1] Threshold ECDSA in Three Rounds. Doerner, Kondi, Lee, Shelat. In S&P 2024.
[2] Trout: Two-Round Threshold ECDSA from Class Groups. Also in ACM CCS 2025.
[3] On the Security of ECDSA with Additive Key Derivation and Presignatures. In EUROCRYPT 2022.

The team

Zengpeng Li[1], Yingjie Lyu[1], Xudong Deng[1], Hong-Sheng Zhou[2],

Mei Wang[1], Puwen Wei[1], Chen Qian[1]

[1] Shandong University @ China

[2] Virginia Commonwealth University @ USA

Structure of ECDSA and the problem of non-linearity

- Elliptic curve group $\mathbb{G}$, generated by g of prime order q
- Secret key $x \leftarrow\$ \mathbb{Z}_q$
- Public key $X := g^x$
- To sign a message whose hash is m:
  1. Sample nonce $k \leftarrow\$ \mathbb{Z}_q$
  2. Compute "presignature" $R := g^k$
  3. Let $r := R|_{\text{x-axis}}$
  4. Compute $\sigma := k^{-1}(m + rx) \bmod q$
  5. Return (r, σ)

Hurdle for MPC: shared secret nonce k inverted then multiplied with shared secret key x

Sketch of the 3-round DKLs24 protocol

1. Jointly sample presignature $R = g^k$ via commit and reveal
2. Let each party obtain additive shares of $k\gamma$ and $x\gamma$ via pairwise 2PC
   (k, x, γ are shared secrets)
3. Right part of signature is assembled as $\sigma = (k\gamma)^{-1} \cdot \gamma(m + rx)$
   (m: message hash; $r = R|_{x\text{-axis}}$)

Towards two-round signing

DKLs24 preprocessing uses 2 rounds for 2 tasks: commit and reveal, share multiplication

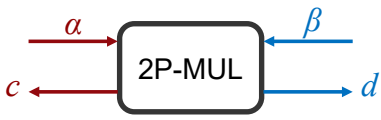
We want to keep the last round fast

How to combine the first 2 rounds into 1?

- Alternative to commit and reveal for distributed generation of R
- Method to multiply shared secrets in 1 round instead of 2

Non-Interactive Multiplication

Practical 1-round (simultaneous 2-message) 2P-MUL is recently achieved [4] [5]



$$\mathrm{m_A} \leftarrow \mathrm{Encode_A}(\alpha)$$

$$\mathrm{m_B} \leftarrow \mathrm{Encode_B}(\beta)$$

$$c \coloneqq \mathrm{Decode_A}(\mathrm{m_B}) \qquad d \coloneqq \mathrm{Decode_B}(\mathrm{m_A})$$

[4] An Algebraic Framework for Silent Preprocessing with Trustless Setup and Active Security. Abram, Damgård, Orlandi, Scholl. In CRYPTO 2022.

[5] Succinct Homomorphic Secret Sharing. Abram, Roy, Scholl. In EUROCRYPT 2024.

Last piece of puzzle: the commitment

Most schemes use “commit and reveal” to distributively generate R

- Each party i commits to g^{k_i} in round 1, and decommits in round 2
- The presignature is $R = \prod_i g^{k_i}$

Simply remove the commitment? Resulting scheme likely insecure

- Nonce distribution can be biased, $\Rightarrow$ attacks
- Incompatible with existing security proof strategy

Removing the commitment

A patch to the construction: re-randomize the presignature in the online phase

- Analogous to FROST[6]
- Allows security reduction to a variant of single-party ECDSA assumption[7]
- Enables immunity to the Groth–Shoup attack surface

[6] FROST: flexible round-optimized Schnorr threshold signatures. Komlo, Goldberg. In SAC 2020.
[7] Two-Round 2PC ECDSA at the Cost of 1 OLE. Adjedj, Blokh, Couteau, Galansky, Joux, Makriyannis. In PKC 2026.

Sketch of our final scheme

In round 1, each party i does:

1. Publish g^{k_i}, aggregate to obtain g^k
2. Input k_i, x_i, γ_i to NIM, obtain shares of $k\gamma$ and $x\gamma$

In round 2:

1. Let z, y be hashes derived from (PK, msg, round-1 transcript)
2. $R = g^{zk+y}$, $r = R|_{x\text{-axis}}$
3. Right part of signature is assembled as $\sigma = \frac{\gamma(m+rx)}{\gamma(zk+y)}$

Implementation and Performance

Protocol implemented in Rust, using BICYCL for class group operations

Experiment results with i7-13700K CPU, single-threaded:

Tool	Scheme	Communication (KiB) with each peer	Computation (ms) for each added party
Paillier OLE	CGGMP20	23	450
Class-group OLE	CCLST20	3.5	120
OLE from OT	DKLs24	50	**10**
Class-group TDec	WMC24	3.4	60
Class-group NIM	Ours	**1.9**	90

Final reference implementation is work in progress

Takeaways

- Two-round threshold ECDSA in the dishonest majority setting
- Fast online phase (<1 ms) and constant-sized protocol messages (1.9 KiB)
- Fixes presigning security via re-randomization

Enabled by:

- Non-Interactive Multiplication from Castagnos–Laguillaumie class group cryptography
- Security of “tweakable” ECDSA presigning

Thank you!

https://eprint.iacr.org/2025/1696