Three-Round (Robust) Threshold ECDSA from Threshold CL Encryption

Bowen Jiang, Guofeng Tang, Haiyang Xue

School of Computing and Information Systems, Singapore Management University

ACISP 2025

Outline

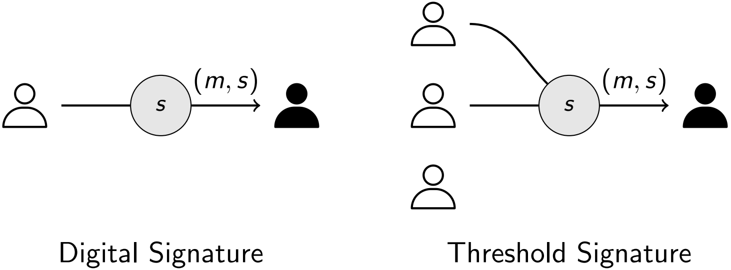
What is & Why Threhsold ECDSA

Our Contributions

Technical Overview

What is Threshold ECDSA?

- Distributed signature generation by a group of parties.
- Requires a $t + 1$ out of n threshold to sign.
- Crucial for security in blockchain systems like Bitcoin and Ethereum.
- Eliminates single points of failure by protecting the private key.



The Core Challenge: Non-Linearity

Thresholding ECDSA is hard because its signing equation is non-linear.

ECDSA Signature Equation

$$s = k^{-1}(H(m) + xr) \pmod q$$

- It involves the **inverse** of a secret (k^{-1}).
- And the **multiplication** of two secrets ($k^{-1} \cdot x$).

- Performing these operations on secret shares requires complex and expensive multi-party computation (MPC) protocols.

Motivation: A Gap in Existing Research

Current state-of-the-art solutions present a trade-off: you can have less rounds or low communication cost, but not both.

Fast but Costly: DKLs24

- **Rounds:** 3 (Fast)
- **Communication:** $O(n)$

Efficient but Slow: WMC24

- **Rounds:** 4 (Slow)
- **Communication:** $O(1)$

Our Question: Can we achieve 3 rounds **and** constant communication?

Our Contributions

We introduce two 3-round schemes based on threshold CL encryption.

TECDSA-Normal

- Basic, non-robust scheme.
- **3 Rounds**.
- $O(1)$ **Communication**.

TECDSA-Robust

- Enhanced with robustness.
- Also **3 Rounds**.
- Sacrifices constant communication for robustness ($O(t)$).

- Implemented in C++ using the BICYCL library.

Comparison of Protocols

Table: Our schemes achieve 3 rounds with better communication trade-offs.

Signing Protocols	Rounds	Computation	Communication	Fault ID	Robust
CCL23[2]	7	$O(n) + O(n^2)$	$O(n) + O(n)$	✓	×
WMY23[5]	6	$O(n) + O(tn^2)$	$O(n) + O(n)$	✓	✓
WMC24[4]	4	$O(n) + O(n)$	$\boldsymbol{O(1)}$	✓	✓
DKLs24[3]	**3**	$O(n)$	$O(n)$	×	×
TECDSA-Normal	**3**	$O(n) + O(n)$	$\boldsymbol{O(1)}$	✓	×
TECDSA-Robust	**3**	$O(n) + O(tn)$	$O(t)$	✓	✓

Fault Identification (Fault ID): To identify misbehaving participants and discard their contributions.
"+" denotes the extra cost for fault identification.

Technical Overview: TECDSA-Normal

The Goal: A "Blinded" ECDSA Signature Form

We transform the standard equation using a blinding factor ϕ:

$$s = \frac{H(m) + xr}{k} \xrightarrow{\text{blind}} s = \frac{\phi(H(m) + xr)}{\phi k}$$

Our goal is to:

- Compute the public nonce R in first two rounds.
- Compute the blinded numerator and denominator in encrypted form.

Technical Overview: TECDSA-Normal

Round 1 (Offline): Generate Secrets

- Parties generate shares of nonce k_i and blinding factor ϕ_i.
- Exchange encrypted values $Enc(\phi_i)$ and commitments $Com(g^{k_i})$.

Status: $Enc(\phi_i) \Rightarrow Enc(\phi)$: Parties homomorphically compute the blinding factor.

Technical Overview: TECDSA-Normal

Round 2 (Offline): Build Encrypted Components

- $Enc(\phi k_i) \Rightarrow Enc(\phi k)$: Homomorphically compute the encrypted denominator $Enc(\phi k)$.
- $Enc(\phi x_i) \Rightarrow Enc(\phi x)$: Compute the message-independent part of the numerator, $Enc(\phi x)$.
- Open commitments to reveal public nonce R (and its x-coordinate r).

Formula after Round 2:

$$s_{enc} = \frac{\overbrace{Enc(\quad ? \quad)}^{\text{Depends on } m} \oplus Enc(\phi x r)}{Enc(\phi k)}$$

*$\oplus$ denotes homomorphic addition on ciphertexts.

Technical Overview: TECDSA-Normal

Final Round (Online): Assemble and Decrypt

- With message m, parties locally compute $Enc(\phi H(m))$.
- This completes the numerator ciphertext.
- Then, perform a threshold decryption.

Formula in Final Round:

$$s_{enc} = \frac{Enc(\phi H(m)) \oplus Enc(\phi x r)}{Enc(\phi k)} = \frac{Enc\big(\phi(H(m) + xr)\big)}{Enc(\phi k)}$$

*$\oplus$ denotes homomorphic addition on ciphertexts.

Technical Overview: TECDSA-Normal

Final Round (Online): Assemble and Decrypt

- With message m, parties locally compute $Enc(\phi H(m))$.
- This completes the numerator ciphertext.
- Then, perform CL threshold decryption twice to obtain $\phi(H(m) + xr)$ and ϕk.

Final Decryption Step:

$$s = \frac{\text{t-CL.Dec}\Big(Enc\big(\phi(H(m) + xr)\big)\Big)}{\text{t-CL.Dec}\Big(Enc(\phi k)\Big)} = \frac{\phi(H(m) + xr)}{\phi k}$$

How TECDSA-Robust Works

- **Problem**: The basic scheme isn't robust. If some party P_{i^*} does not contribute correct k_{i^*} in Round 2, the protocol will get $R = g^{k - k_{i^*}}$, where $k = k_1 + \cdots + k_n$.
- **Solution**: We change the nonce sharing k_i from an "all-or-nothing" (n-of-n) scheme to a "fault-tolerant" (t-of-n) threshold structure.
- **Method**: We use a 2-round Distributed Randomness Generation (DRG) protocol [1] with Publicly Verifiable Secret Sharing (PVSS).

This allows honest parties to identify and exclude malicious actors and reconstruct the correct nonce.

Crucially, the DRG runs simultaneously with the first two rounds of TECDSA-Normal, **so no extra rounds are added!**

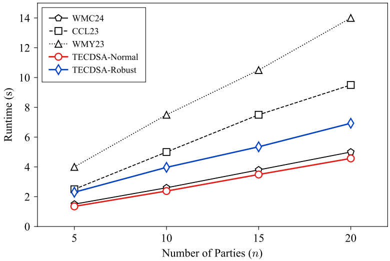
Implementation & Setup

- **Library**: Using an C++ library BICYCL optimized for class group arithmetic operations.
- **Availability**: Code is public on Github repositories.
- **Configuration**: 128-bit security level.

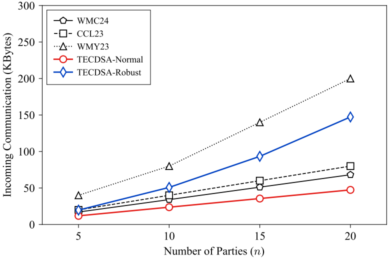
Signing Protocols	(1, 2)-threshold		(2, 3)-threshold		(4, 5)-threshold	
	Time	Out.Comm	Time	Out.Comm	Time	Out.Comm
CCL23 [2]	1012	6553	1518	10752	2531	19148
WMY23 [5]	1730	7680	2595	7850	4328	8192
WMC24 [4]	631	3490	938	3490	1564	3490
TECDSA-Normal	540	2426	810	2426	1350	2426
TECDSA-Robust	991	3330	1284	3564	1811	4032

Table: Runtime of total protocol and communication overhead of offline phase per party under small-scale setting.

Implement and Comparison



(a) Runtime per party



(b) Incoming communication per party

Figure: Runtime of total protocol and (incoming) communication overhead of offline phase under threshold setting $t = n - 1$.

Performance Highlights

Our schemes show significant efficiency improvements.

TECDSA-Normal (Basic Scheme)

- Achieves the best performance among all compared protocols.
- Fastest runtime and lowest communication overhead.

TECDSA-Robust (Robust Scheme)

- Reduces execution time by 50% and communication by 23% compared to WMY23 (at $n = 20$).
- Has one fewer round than WMC24, making it better for high-latency networks.

Conclusion & Future Work

- We designed threshold ECDSA protocols that are both **fast (3 rounds)** and **communication-efficient**.
- Our **TECDSA-Normal** scheme is the first to achieve 3-round signing with constant communication, offering the best performance.
- Our **TECDSA-Robust** scheme provides a 3-round robust alternative that has less rounds than prior work.
- This work provides a clear choice for practitioners:
  - Choose **Normal** for maximum performance.
  - Choose **Robust** for guaranteed liveness in hostile environments.
- Future work includes designing fewer-round threshold ECDSA, improving MPC and MtA constructions, optimizing zero-knowledge proofs, and reducing online/offline cost.

References I

- Cascudo, I., David, B.: Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In: ASIACRYPT 2024. pp. 216–248. Springer (2024)
- Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold ec-dsa revisited: Online/offline extensions, identifiable aborts proactive and adaptive security. Theoretical Computer Science **939**, 78–104 (2023)
- Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ecdsa in three rounds. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 3053–3071. IEEE (2024)
- Wong, H.W., Ma, J.P., Chow, S.S.: Secure multiparty computation of threshold signatures made more efficient. In: NDSS 2024 (2024)
- Wong, H.W., Ma, J.P., Yin, H.H., Chow, S.S.: Real threshold ecdsa. In: NDSS (2023)

Thank You

Q & A