

“Preview Writeup”: In anticipation of a package submission to the NIST Threshold Call

Title: Menshen and Octopus-(T)CL

Subtitle: Two-party and Threshold ECDSA Schemes

Version: Preview Writeup 1.0 (2026-09-20)¹

Team name: Octopus

Team members: Man Ho Au, Robert H. Deng, Liming Gao, Bowen Jiang, Xianhui Lu, Tian Qiu, Guofeng Tang, Ruida Wang, Haiyang Xue, Guomin Yang, Chengru Zhang

Abstract: This document previews the Octopus team’s proposals for the NIST Multi-Party Threshold Cryptography Call, specifying three distinct 3-round ECDSA schemes: a two-party protocol and two t -out-of- n threshold signature protocols.

First, our two-party ECDSA scheme (denoted as Menshen) requires only a single multiplicative-to-additive (MtA) conversion, also known as Oblivious Linear Evaluation (OLE). Second, we propose two variants of t -out-of- n ECDSA schemes: Octopus-CL, which is built upon standard Castagnos-Laguillaumie encryption, and Octopus-TCL, which leverages threshold Castagnos-Laguillaumie encryption.

Proposed crypto-systems:

- (I) Menshen: Two-party ECDSA (Categories N1)
- (II) Octopus-CL: t -out-of- n ECDSA (Categories N1)
- (III) Octopus-TCL: t -out-of- n ECDSA (Categories N1)

Keywords: Threshold Cryptography; NIST Threshold Call; Two-party ECDSA; t -out-of- n ECDSA



Document License: [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

¹Version submitted to NIST-MPTC for publication

Preview writeup. This document is provided to NIST for online publication, to foster public awareness and support public discussion within the scope of the NIST First Call for Multi-Party Threshold Schemes [NIST-IR8214C]. This “preview writeup” represents a good-faith plan for a subsequent “package submission”. However, until the deadline for package submission, the team may still modify its own composition and the submission plan, including possible changes to the technical scope, and/or the used techniques or achieved results.

Team members: Man Ho Au ^{i1,a1}, Robert H. Deng ^{i2,a3}, Liming Gao ^{i3,a2*}, Bowen Jiang ^{i4,a3*}, Xianhui Lu ^{i5,a2}, Tian Qiu ^{i6,a4}, Guofeng Tang ^{i7,a3}, Ruida Wang ^{i8,a2}, Haiyang Xue ^{i9,a3}, Guomin Yang ^{i10,a3}, Chengru Zhang ^{i11,a5}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0003-2068-9530); i2 (0000-0003-3491-8146); i3 (0009-0004-0211-4148); i4 (0009-0002-8867-866X); i5 (0000-0001-7091-5810); i6 (0009-0006-2429-8771); i7 (0000-0002-0121-5471); i8 (0009-0002-1276-1299); i9 (0000-0002-0173-7894); i10 (0000-0002-4949-7738); i11 (0009-0001-8755-5206)

Affiliations:

^{a1} Department of Computing, The Hong Kong Polytechnic University

^{a2} Institute of Information Engineering, Chinese Academy of Sciences

^{a3} SCIS, Singapore Management University

^{a4} Digital Trust Centre, Nanyang Technological University

^{a5} Ethereum Foundation

Associateship clarifications:

* Ph.D student (non-employee).

Main contacts:

- **Team mailing list:** octopus_mptc@googlegroups.com
- **Primary technical contact person:** Haiyang Xue, haiyangxue@smu.edu.sg
- **Secondary contact person 1:** Guofeng Tang, tang.guofeng789@gmail.com
- **Secondary contact person 2:** Bowen Jiang, bowen.jiang.2024@phdcs.smu.edu.sg
- **Secondary contact person 3:** Chengru Zhang, chengru.zhang@ethereum.org

Produced by humans. The team hereby confirms that the content in this preview writeup: (i) was produced by the team members, and (ii) was not produced by generative artificial intelligence (AI), with the possible exception of AI-proposed grammar improvements, minor integrated suggestions, or some well-identified and short localized portions of auxiliary content (e.g., some illustration); and (iii) was proofread by the team members.

1. Introduction

The scope of this preview document is to present our plan for submitting two-party and general t -out-of- n threshold ECDSA schemes.

We plan to propose three schemes: 1) Menshen, a two-party threshold ECDSA from only one multiplicative-to-additive conversion (MtA), 2) Octopus-CL, a t -out-of- n ECDSA from standard Castagnos-Laguillaumie (CL) encryption, and 3) Octopus-TCL, a t -out-of- n ECDSA based on threshold Castagnos-Laguillaumie encryption.

All three submissions focus on thresholding NIST-standard ECDSA signatures [FIPS-186-5, §6] that fall under Category N1: Signing, Subtype: QV, and Specification: ECDSA Sign of the NIST Threshold Call [NIST-IR8214C].

2. Specification

Organization. Our final submission will feature three crypto-systems: the first focuses on a two-party setting, while the remaining two provide a t -out-of- n access structure. We briefly introduce each crypto-system in the following subsections.

All three cryptosystems include a distributed key generation phase executed once, followed by a signing phase that may be run multiple times as required.

2.1. Menshen

Menshen is a revisited two-party threshold ECDSA signing protocol based on our team’s paper [XAXYC21]. The key generation phase is standard, relying on Shamir’s secret sharing. The signing process consists of a two-round offline phase, which is independent of the message, and a single-round online phase that depends on the message.

2.1.1. System model

The threshold profile is 2-out-of- n , corresponding to a two-threshold ECDSA setting in the literature. Key generation is executed among any $n \geq 2$ parties, after which any 2 of those n parties can cooperatively generate a valid signature.

The parties operate within a synchronous network and are assumed to be connected by authenticated point-to-point channels. We do not assume the availability of a broadcast channel.

The primary and most expensive building block is Multiplication-to-Addition (MtA) conversion, also known as Oblivious Linear Evaluation (OLE) in the literature. This underlying MtA can be instantiated using various primitives, such as Castagnos-Laguillaumie [CL15], Paillier [Pai99; LN18; XLXADY24], Joye-Libert encryption [JL13; XALCCXYZ23], as well as Oblivious Transfer [DKLS18],

offering a diverse range of trade-offs in computation, communication, setup requirements, and underlying assumptions.

2.1.2. Key idea

Our aim is to build two-party ECDSA from a single MtA call, and to make that call *key-independent*: neither of its inputs is a share of the signing key. Key-independence is what allows the MtA to be preprocessed offline, batched, and reused across keys.

We share the signing key additively and the nonce *linearly*:

$$x = x_1 + x_2, \quad k = k_1(r_1 + k_2),$$

where k_1, r_1 are chosen by P_1 and k_2 by P_2 . The one MtA call takes a fresh $x'_1 \leftarrow \mathbb{Z}_q$ from P_1 and the nonce share k_2 from P_2 , and returns $t_A + t_B = x'_1 k_2$. Party P_1 then sends a single masked value $cc = t_A + x'_1 r_1 - x_1$, and P_2 sets $x'_2 = x_2 - (t_B + cc)$. The key is now re-shared as

$$x = x'_1(r_1 + k_2) + x'_2,$$

which is *aligned with the nonce*: the factor $(r_1 + k_2)$ appears in both. Signing therefore needs no further interaction. P_2 sends $s_2 = (r_1 + k_2)^{-1}(H(m) + rx'_2)$ and P_1 outputs $s = k_1^{-1}(s_2 + rx'_1)$.

Note that the signing key never enters the MtA; it appears only inside the masked value cc . The role of r_1 is to keep it hidden there. Were r_1 absent, a malicious P_2 could feed $k_2 = 0$ into the MtA and read off $x_1 = -t_B - cc$. Because P_1 samples r_1 only after k_2 has been committed, $r_1 + k_2 = 0$ occurs with probability at most $1/q$, so no zero-knowledge proof that $k_2 \neq 0$ is required. The resulting protocol runs in three passes with one offline MtA, and its online phase costs one field element plus a signature verification.

2.1.3. Security

Menshen supports any $n \geq 2$ parties in the key generation, and provides security against a static adversary corrupting up to $f = 1$ party. Menshen is proven secure in the random oracle model in the presence of a malicious static adversary under the real/ideal definition. The assumptions are required based on different instantiations of Multiplication-to-Addition (MtA) conversion.

2.2. Octopus-CL

Octopus-CL is a t -out-of- n threshold ECDSA protocol based on our team's work [TX25]. The signing phase consists of a two-round offline phase and a one-round online phase.

2.2.1. System model

The threshold profile is t -out-of- n . Key generation is executed among any n parties, after which any t of those n parties can cooperatively generate a valid signature.

The parties operate within a synchronous network and are assumed to be connected by authenticated point-to-point channels and a broadcast channel.

Octopus-CL relies on CL encryption [CL15] and its associated zero-knowledge proofs. Its security is based on the hard subgroup membership (HSM) assumption and the smoothness of CL encryption over class groups.

2.2.2. Key Idea

Our aim is a threshold ECDSA that is robust and identifies cheaters, yet still runs in three rounds with a cheap online phase. Writing the signature as

$$s = \frac{\gamma(m + rx)}{\gamma k}$$

for a temporary nonce γ , the usual route to robustness is to t -threshold-share γ , so that any t parties can reconstruct it. Our first observation is that this is unnecessary: γ occurs in both the numerator and the denominator and cancels in the final assembly, so its value need never be pinned down. We therefore let each party keep γ_i as a plain *additive* share of an *undetermined* γ , whose contributors and value are fixed only in the last round, after cheaters have been excluded. Removing a cheater merely replaces γ by $\gamma - \gamma_{i^*}$, which cancels just as well.

Two further points make this work. First, cheater identification must use *public* checking. We show that MtA with private checking becomes insecure once robustness is added: forcing an honest Alice to reveal her private value α for identification lets an adversary holding $t - 1$ shares recover her nonce share, hence k , and hence the signing key from the signature that the robust protocol goes on to produce. We therefore build a CL-based multi-party MtA whose messages are all publicly verifiable, so cheaters are excluded without any complaint or dispute phase, and we aggregate the per-pair proofs into a single constant-size proof with batch proving and verification. Second, because γ is undetermined, there is no public value such as γG against which to check its consistency between numerator and denominator. Instead, each party publishes $\gamma_i R$ and $\gamma_i(mG + rX)$, which stand in a DDH relation, and proves that relation with a standard DDH-ZKP over the elliptic-curve group. This replaces the Paillier- or CL-based proofs of prior work with a handful of EC operations, and is what makes our online phase two to three orders of magnitude cheaper than threshold-decryption-based designs.

2.2.3. Security

Octopus-CL is proven to securely realize the $\mathcal{F}_{\text{ECDSA}}$ functionality against a malicious adversary statically corrupting up to $t - 1$ parties.

Octopus-CL provides cheater identification for any threshold $t \leq n$. In the honest-majority setting, i.e., $n \geq 2t - 1$, Octopus-CL additionally guarantees *robustness*; if cheaters are detected, the protocol isolates them and allows the remaining honest parties to successfully complete the signing process and produce a valid signature, provided their count still meets or exceeds t .

2.3. Octopus-TCL

Octopus-TCL is a t -out-of- n threshold ECDSA protocol based on our team's work [JTX25]. The signing phase also runs with a two-round offline phase and a one-round online phase.

2.3.1. System model

The threshold profile is t -out-of- n . Key generation is executed among any n parties, after which any t of those n parties can cooperatively generate a valid signature. The parties operate within a synchronous network and are assumed to be connected by an authenticated point-to-point channel. We do not assume the availability of a broadcast channel.

2.3.2. Key idea

Our aim is a threshold ECDSA built on threshold CL encryption that keeps the small, near-constant outgoing communication such schemes allow, while cutting the round count from four to three. Earlier threshold-CL designs are forced into four rounds by how they rewrite the signature. Taking $s = k(H(m) + xr)$ with $R = g^{k^{-1}}$ means the public nonce can only be recovered as $(g^\phi)^{(k\phi)^{-1}}$, which requires a threshold decryption of its own. Hence r is unavailable until the third round, and the decryption that yields s must wait for a fourth.

We instead keep ECDSA in its original form,

$$s = k^{-1}(H(m) + xr),$$

and divide the work differently. In Round 1 each party encrypts its share ϕ_i of a blinding value ϕ under the threshold CL key and commits to its nonce point $R_i = g^{k_i}$. In Round 2, with ϕ assembled homomorphically, each party publishes ϕx_i and ϕk_i , together with proofs that the same scalar appears in the ciphertext and in X_i and R_i , and the commitments to R_i are opened. Both R and r are therefore fixed before the message arrives.

The online round is then a single threshold decryption of two ciphertexts, ϕk and $\phi(H(m) + rx)$, the second assembled locally once the message is known. Their quotient is s , with ϕ cancelling.

This basic protocol is not yet robust. Each k_i is an n -out-of- n additive share, so a party that withholds a correct opening of R_i shifts the nonce to $g^{k-k_i^*}$ and thereby biases R . The same gap appears in the security proof, where the simulator can no longer place the oracle's R among the honest parties' contributions.

Our robust variant gives k a t -out-of- n sharing whose shares are publicly verifiable, so a misbehaving party is identified and excluded while the remaining parties still determine the same R . We obtain this from a two-round distributed randomness generation over class groups, which runs concurrently with the two offline rounds and so costs no extra round; the same substitution makes the distributed key generation robust, replacing verifiable secret sharing with its publicly verifiable counterpart. The price is that outgoing communication is no longer constant but grows with the threshold, a trade that stays cheap in the small-scale deployments these protocols typically target.

2.3.3. Security

Octopus-TCL relies on a threshold version of CL encryption [CL15] and associated zero-knowledge proof of knowledge [BCL25]. Octopus-TCL achieves security with abort, and is proven to securely realize $\mathcal{F}_{\text{ECDSA}}$ in the random oracle model in the presence of a malicious static adversary under the real/ideal definition, under the assumption of IND-CPA of the threshold CL encryption.

3. Open-Source Implementation

1. **Code structure:** Our reference implementation will provide complete, testable Rust implementations of Menshen, Octopus-CL, and Octopus-TCL in a single, layered codebase. The primitive layer will contain core types, curve operations, and reusable primitives. It will include a CL library shared by all three schemes and implemented in Rust using rug and FFI bindings to GMP. The protocol layer will contain separate state machines for each scheme. The networking layer will handle message encoding, authenticated point-to-point and broadcast communication, and session management for these state machines. The top layer will expose the public API and provide tools for execution, testing, and benchmarking. The code will run on the NIST baseline platform with pinned toolchain and dependency versions.
2. **Code progress and availability:** The core Rust framework and implementations of all three schemes are already in place in a private Git repository. We will refine the codebase to meet the NIST requirements and audit each implementation against its protocol specification. At submission, the repository will be made public at a fixed commit with instructions for building, running, testing, and benchmarking the code.
3. **Implementation of the networking model:** Our schemes use authenticated point-to-point channels and/or broadcast channels. Each party will run as a separate process and communicate with its peers over TCP. Point-to-point messages will be signed by their senders and bound to the protocol and session in which they are sent. Broadcast will be implemented over these authenticated connections to provide the consistency or agreement guarantees required by each scheme. Configuration files will specify party identities, public keys, network addresses, threshold parameters, and session information.
4. **Testing:** We will provide a test suite with two tiers. The deterministic tier uses fixed inputs and recorded randomness to make complete executions reproducible. It includes

known-answer vectors and explicit edge cases. The randomized tier samples randomness freshly to provide better coverage, allowing us to find bugs that the deterministic tier does not catch. We will introduce malicious deviations and network disruptions at different protocol stages, and verify that honest parties either complete correctly or terminate without accepting invalid results. Cheater identification and robustness will be tested where provided by the scheme.

4. Licensing, Patent Claims, and Funding

1. **Licensing:** The core Rust code will be released under MIT OR Apache-2.0. All Bundled Dependencies and External Dependencies will use licenses as required by the NIST Threshold Call. The package will record licenses using SPDX expressions, preserve required notices, identify the license applicable to each component, and verify that the compiled reference implementation can be redistributed under an allowed license.
2. **Patent claims:** To our knowledge, no patent has or could have claims covering the contents of our submission.

References

- [FIPS-186-5] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. (U.S. Department of Commerce) *Federal Information Processing Standards (FIPS) 186-5*. February 2023. DOI: [10.6028/NIST.FIPS.186-5](https://doi.org/10.6028/NIST.FIPS.186-5).
- [BCL25] Agathe Beaugrand, Guilhem Castagnos, and Fabien Laguillaumie. “Efficient succinct zero-knowledge arguments in the CL framework”. In: *Journal of Cryptology* 38.1 (2025), p. 13. DOI: [10.1007/s00145-024-09534-1](https://doi.org/10.1007/s00145-024-09534-1). URL: <https://eprint.iacr.org/2024/1966>.
- [CL15] Guilhem Castagnos and Fabien Laguillaumie. “Linearly homomorphic encryption from DDH”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2015, pp. 487–505. DOI: [10.1007/978-3-319-16715-2_26](https://doi.org/10.1007/978-3-319-16715-2_26). URL: <https://eprint.iacr.org/2015/047>.
- [DKLS18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. “Secure two-party threshold ECDSA from ECDSA assumptions”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 980–997. DOI: [10.1109/SP.2018.00036](https://doi.org/10.1109/SP.2018.00036). URL: <https://eprint.iacr.org/2018/499>.
- [JL13] Marc Joye and Benoît Libert. “Efficient cryptosystems from 2 k-th power residue symbols”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2013, pp. 76–92. DOI: [10.1007/978-3-642-38348-9_5](https://doi.org/10.1007/978-3-642-38348-9_5). URL: <https://eprint.iacr.org/2013/435.pdf>.
- [JTX25] Bowen Jiang, Guofeng Tang, and Haiyang Xue. “Three-Round (Robust) Threshold ECDSA from Threshold CL Encryption”. In: *Australasian Conference on Information Security and Privacy*. Springer. 2025, pp. 224–244. DOI: [10.1007/978-981-96-9095-4_12](https://doi.org/10.1007/978-981-96-9095-4_12). URL: <https://eprint.iacr.org/2026/190>.
- [LN18] Yehuda Lindell and Ariel Nof. “Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1837–1854. DOI: [10.1145/3243734.3243788](https://doi.org/10.1145/3243734.3243788). URL: <https://eprint.iacr.org/2018/987>.
- [Pai99] Pascal Paillier. “Public-key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology – EUROCRYPT ’99*. Springer. 1999, pp. 223–238. DOI: [10.1007/3-540-48910-X_16](https://doi.org/10.1007/3-540-48910-X_16).
- [TX25] Guofeng Tang and Haiyang Xue. “Robust Threshold ECDSA with Online-Friendly Design in Three Rounds”. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2025, pp. 203–221. DOI: [10.1109/SP61157.2025.00115](https://doi.org/10.1109/SP61157.2025.00115). URL: <https://eprint.iacr.org/2025/910>.
- [XALCCXYZ23] Haiyang Xue, Man Ho Au, Mengling Liu, Kwan Yin Chan, Handong Cui, Xiang Xie, Tsz Hon Yuen, and Chengru Zhang. “Efficient multiplicative-to-additive function from Joye-Libert cryptosystem and its application to threshold ECDSA”. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2023, pp. 2974–2988. DOI: [10.1145/3576915.3616595](https://doi.org/10.1145/3576915.3616595). URL: <https://eprint.iacr.org/2023/1312>.

[XAXYC21] Haiyang Xue, Man Ho Au, Xiang Xie, Tsz Hon Yuen, and Handong Cui. “Efficient online-friendly two-party ECDSA signature”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 558–573. DOI: [10.1145/3460120.3484803](https://doi.org/10.1145/3460120.3484803). URL: <https://eprint.iacr.org/2022/318>.

[XLXADY24] Zhikang Xie, Mengling Liu, Haiyang Xue, Man Ho Au, Robert H Deng, and Siu-Ming Yiu. “Direct Range Proofs for Paillier Cryptosystem and Their Applications”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 899–913. DOI: [10.1145/3658644.3690261](https://doi.org/10.1145/3658644.3690261). URL: <https://eprint.iacr.org/2024/1355>.

[NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2026. DOI: [10.6028/NIST.IR.8214C](https://doi.org/10.6028/NIST.IR.8214C).