

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

Title: TORTL DKG

Subtitle: Two-Round Trustless Distributed Key Generation

Version: Preview Writeup 0.2 (2026-09-21)¹

Team name: Kelvin Zero

Team members: Maxime Godon, Florian Le Mouël, Nadège Pulgar-Vidal, Thierry St-Jacques-Gagnon

Abstract: This document outlines members of the Two-Round Trustless Distributed Key Generation (TORTL DKG) family of cryptosystems, intended for submission to the NIST Multi-Party Threshold Cryptography (MPTC) call. More specifically, TORTL-EC-P256 DKG, the submitted elliptic curve member of the family, is based on Shamir’s Secret Sharing Scheme (SSSS). It removes the trusted third party at the expense of an extra communication round and a verification step based on prior work by Agrawal et al. CCS’18. This submission also introduces TORTL-ML DKG, a lattice-based DKG that minimizes exchanged data size via functional verification. It leverages standard building blocks, such as Replicated Secret Sharing (RSS) and quantum-adapted Chaum-Pedersen-type Zero-Knowledge Proofs (ZKP), to construct a trustless, two-round DKG with exact reconstruction, verifiable correctness, and secrecy. Family members offer proven security against malicious adversaries with minimal communication rounds. Furthermore, their highly adaptable design allows them to serve as a DKG layer for other threshold schemes such as Trout++, Twig, KU, and Mithril.

Proposed crypto-systems: (I) TORTL-EC-P256 DKG: Two-Round Trustless Elliptic Curve P-256 Distributed Key Generation (Categories N4.1); (II) TORTL-ML: Two-Round Trustless Module-Lattice Distributed Key Generation (Categories N4.3).

Keywords: Threshold Cryptography; NIST Threshold Call; DKG; Secret Sharing; Commitment schemes; ZKP; Lightweight



(Logo AI-generated with Gemini Images)

¹Preliminary version submitted to NIST-MPTC for review

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: Maxime Godon ^{i1,a2,†}, Florian Le Mouël ^{i2,a1,*}, Nadège Pulgar-Vidal ^{i3,a2}, Thierry St-Jacques-Gagnon ^{i4,a2}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0009-0006-1091-5345); i2 (0009-0003-2192-7462); i3 (0009-0008-9424-7421); i4 (0009-0002-5631-3631)

Affiliations:

^{a1} Département de GIGL, [Polytechnique Montréal](#) @ Montréal, Québec, Canada

^{a2} Kelvin Zero Inc. @ Montréal, Québec, Canada

Associateship clarifications:

* Ph.D. work performed as a student. † Contractor via Serenest Cybersecurity Inc.

Main contacts:

- **Team mailing list:** tortl@googlegroups.com
- **Primary technical contact person:** Thierry St-Jacques-Gagnon, teegagnon+tortl@gmail.com
- **Secondary contact person 1:** Florian Le Mouël, florian.lmouel@gmail.com
- **Secondary contact person 2:** Maxime Godon, maxime.godon@gmail.com

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

In this work, we describe TORTL DKG: a Two-Round Trustless Distributed Key Generation family of cryptosystems. Specifically, we present the TORTL-EC (Elliptic Curve) P-256 DKG and TORTL-ML (Module Lattice) DKG members of this family. This submission fits within Categories N4.1 and Category N4.3 for TORTL-EC-P256 DKG and TORTL-ML DKG respectively. Our goal is to provide an efficient trustless and dealerless distributed key generation (DKG) system that is modular and can easily be included in other schemes, whether in a Quantum-Vulnerable (QV) or Post-Quantum (PQ) setting.

This DKG scheme was first defined for Trustless Distributed Symmetric Encryption (T-DiSE) [MGBBBC24], which was developed by Kelvin Zero Inc., the company sponsoring this project. It has since been used at KZero Passwordless and Trustee, two companies that integrated it into their products. The DKG part of the T-DiSE code, which used the secp256k1 curve, has been adapted to be a member of the TORTL DKG family as TORTL-EC secp256k1 DKG, or TORTL-EC-K1 DKG. As such, we will use TORTL-EC DKG to refer to the TORTL DKG members using elliptic curves.

However, note that we are not submitting TORTL-EC-K1 DKG for this call as we saw more value in supporting NIST-approved curves. Nevertheless, it demonstrates the flexibility and agility of the TORTL DKG family.

In part due to this modularity, TORTL DKG members can be used by several other submitted schemes, including Trout++ [DNP26], Twig [LLDZWWQ26], and KU [KUVS26], which currently lack a DKG. TORTL-ML can be adapted to different ring parameters, and we believe that it is compatible with the submitted Mithril [CDPENP26] signature scheme.

For TORTL-EC DKG, we leverage classic schemes such as Shamir's Secret Sharing Scheme (SSSS) [Sha79], which we adapt to be dealerless, along with Pedersen commitments [Ped91b] and the Chaum-Pedersen protocol [CP92]. Our scheme is secure in the Random Oracle Model (ROM) under the Decisional Diffie-Hellman (DDH) assumption. With regard to TORTL-ML, we leverage lattice-based advances in the field, and our scheme is secure in the Random Oracle Model (ROM) under the Module Learning With Errors (MLWE) and the Module Short Integer Solution (MSIS) assumptions. For a participant threshold t and a number of parties n , both of our schemes are secure for up to $t - 1$ static malicious corruptions, with $t \leq n$.

In TORTL-EC DKG, the size of each message is constant, but for TORTL-ML DKG the size of our messages increases exponentially with t and n .

Finally, our schemes are trustless, require only two rounds of communication, and achieve security with unanimous abort while preserving correctness and secrecy. All parties are guaranteed that the DKG either successfully establishes a key or completely aborts without identifying individual malicious actors. A typical complaint-based DKG, such as the Pedersen DKG [Ped91a], uses a large number of commitments to verify each step of the key generation and exclude malicious contributions to guarantee robustness. By leveraging functional verification of the secret sharing, we significantly reduce the number of commitments necessary, thus reducing the bandwidth of the DKG.

2. Specification

2.1. Organization

The technical specification documents the TORTL DKG family of cryptosystems. We detail the public parameters followed by explanations of share generation, Distributed Verifiable Random Function (DVRF) computation, and functional verifications for selected TORTL DKG members.

All TORTL DKG members include a secret sharing scheme, a structured set over which to perform the secret sharing, a DVRF (including a commitment scheme used to commit to secret shares and a Zero-Knowledge Proof (ZKP) used to verify evaluations performed with a secret share), and a hash function to a structured set element. Although all of these components are modular, we prefer swapping the secret sharing scheme, the structured set, and the DVRF to produce new members, while keeping the same blueprint. The hash function used to map to the structured set depends on the choice of structured set. We believe this modularity makes TORTL DKG widely compatible and easily integrated with other proposed schemes and protocols.

TORTL-EC-P256 DKG uses SSSS [Sha79], the NIST P-256 elliptic curve, and the DVRF from [AMMR18] (equipped with the Pedersen commitment [Ped91b] and the Chaum-Pedersen ZKP [CP92]); TORTL-ML DKG uses Replicated Secret Sharing (RSS), a custom ring ensuring both MLWE and MSIS are hard, and the DVRF from [EKSSZSC21] (equipped with the Baum-Damgard-Lyubashevsky-Oechsner-Peikert (BDLOP) commitment [BDLOP18] and a ZKP inspired by Lyubashevsky’s identification scheme [Lyu09] and the Chaum-Pedersen ZKP [CP92]).

2.2. System Model

We propose a DKG for a threshold t and n parties (t, n in $\mathbb{Z}^+$, $t \leq n$), among which $t - 1$ may behave as Byzantine adversaries. We operate in a static corruption model; as such, the corrupted parties are not allowed to change within a single setup of the protocol. While TORTL-EC DKG has no restrictions on the maximum values of t and n , in the context of TORTL-ML DKG, the size of messages increases exponentially with t and n . As such, our PQ scheme is unable to remain competitive for a high number of parties. Our current experimental result, yielding message sizes under 60 kB, is achieved with $t = 4$ and $n = 5$.

The protocol operates in two rounds. Observe that the notion of rounds exists from the perspective of individual parties only and is not a shared or synchronized state. As such, it is possible and supported for a given party to consider itself in the first round while a different party considers itself in the second round, which removes the need for a centralized coordinator. Participants may also engage in several instances of the protocol in parallel using different keys for each, whether with the same participant set or not. Each participant would track the state of the setup and the keys used for each protocol instance they participate in. The protocol is also designed to be fail-safe, meaning that it can be interrupted unilaterally, at any time, by any party. As soon as a malicious behaviour is detected, the protocol execution is aborted for safety, ensuring compromised nodes cannot silently corrupt the key generation process.

2.2.1. Public Parameters

- The safety parameter, $\lambda = 128$
- The number of parties, n
- The participant threshold of the scheme, t , and the correlated maximum number of corrupted parties, $t - 1$. We require $1 < t \leq n$.
- The identifier of a party is a positive non-zero integer, denoted by i or j .

TORTL-EC DKG parameters:

- A cyclic group G over $\mathbb{Z}_p$ of prime order q s.t. $q|p - 1$ and the Discrete Logarithm (DLOG) problem is hard in this group (in practice, we use elliptic curve groups).
- Two generators of G , g_1 and g_2 , such that $\log_{g_1}(g_2)$ is unknown.
- A hash function, $h()$, along with Domain Separation Tags (DST) to separate its various usages.
- A function $f()$ that maps the output of the hash function to $\mathbb{Z}_p$
- h' , an element of G

TORTL-ML DKG parameters:

- A ring $R_q = \mathbb{Z}_q[X]/(X^d + 1)$
- A subring $\hat{R}_p = \mathbb{Z}_p[X]/(X^{d/s} + 1)$
- The Hamming weight of the challenge, κ
- A and B , two public matrices of dimension m by k composed of elements in R_q .
- A public label μ
- b , a vector of elements of $\hat{R}_p^k$
- Three hash functions, $H_1()$, $H_2()$ and $H_3()$, used to generate the Fiat-Shamir challenge $chal$ (for H_1), A and B (for H_2) and b (for H_3).
- The rejection ceiling β
- $S = \{I \text{ s.t. } |I| = n - t + 1\}$ the set of parties receiving each key.

2.2.2. First Round

During the first round, each party acts as a dealer for a *partial secret*. Each party will therefore communicate a *partial secret share* to all parties including itself.

TORTL-EC DKG first round:

Each party samples a degree $t - 1$ polynomial (in some global variable Z) whose coefficients are elements of $\mathbb{Z}_q$ and evaluates this polynomial at $Z = i$ for each party. Each party then sends to each other party the resulting value. Call this value the partial secret share $psk_{j \rightarrow i}$ when it is calculated by participant j and sent to participant i .

TORTL-ML DKG first round:

Each party samples uniform ternary $a_I \in R_q^k$ for each $I \in S$. Each party then sends to each party in I the value a_I . Call this value the partial secret share $psk_{j \rightarrow I}$ when it is calculated by participant j and sent to participants in I .

2.2.3. Second Round

The second round acts as a layer of verification to ensure honesty among all parties. A party i enters the second round once it has received all $psk_{\dots \rightarrow i} / psk_{\dots \rightarrow I}$.

TORTL-EC DKG second round:

- Sum up all received partial secret shares to form a secret share. Call this sk_i .
- Calculate $com_i = sk_i \cdot g_1 + r_i \cdot g_2$ for some secret randomness $r_i \in \mathbb{Z}_q$
- Calculate $eval_i = sk_i \cdot h' \in G$
- Sample y_1, y_2 uniformly from $\mathbb{Z}_q$
- Calculate $t_{1,i} = y_1 \cdot h'$ and $t_{2,i} = y_1 \cdot g_1 + y_2 \cdot g_2$
- Calculate $chal_i = h(h', eval_i, com_i, g_1, g_2, t_{1,i}, t_{2,i})$
- Calculate $z_{1,i} = (y_1 - chal_i \cdot sk_i) \bmod q$ and $z_{2,i} = (y_2 - chal_i \cdot r_i) \bmod q$
- Broadcast $(com_i, eval_i, chal_i, \pi_i = (z_{i,1}, z_{i,2}))$

TORTL-ML DKG second round:

- Verify that $\|psk\|_\infty \leq 1$ for all received psk
- Sum up all received partial secret shares to form a secret share. Call this sk_I .
- Calculate $com_{i,I} = A \cdot sk_I + B \cdot r_{i,I}$ for some ternary vector $r_{i,I} \in R_q^k$
- Calculate $eval_{i,I} = \langle b, sk_I \rangle \in \hat{R}_p$
- Sample $y_{i,I,1}$ and $y_{i,I,2}$ uniformly in $[-\beta, \beta]^k$
- Calculate $t_{i,I,1} = \langle b, y_{i,I,1} \rangle$, $t_{i,I,2} = A \cdot y_{i,I,1} + B \cdot y_{i,I,2}$
- Calculate $chal_{i,I} = H_1(b, eval_{i,I}, com_{i,I}, A, B, t_{i,I,1}, t_{i,I,2})$
- Calculate $z_{i,I,1} = y_{i,I,1} - chal_{i,I} \cdot sk_I$, $z_{i,I,2} = y_{i,I,2} - chal_{i,I} \cdot r_{i,I}$, rejection-sample until $\|z_{i,I,1}\|_\infty$ and $\|z_{i,I,2}\|_\infty \leq \beta - \kappa \cdot n$
- Broadcast $(com_{i,I}, eval_{i,I}, chal_{i,I}, \pi_{i,I} = (z_{i,I,1}, z_{i,I,2}))$ for each I that contains i .

2.2.4. Verification

After receiving the output of the second round from all other parties, each party i will check the proofs submitted by each other party:

TORTL-EC DKG verification:

- Recalculate $t'_{1,i} = z_{1,i} \cdot h' + com_i \cdot eval_i$ and $t'_{2,i} = z_{1,i} \cdot g_1 + z_{2,i} \cdot g_2 + chal_i \cdot com_i$
- Confirm $chal_i = h(h', eval_i, com_i, g_1, g_2, t'_{1,i}, t'_{2,i})$

Finally, perform the following functional checks:

Define $L(Set)$ as the Lagrange interpolation of all evaluations $eval_i$, for $i \in Set$. Observe that for Set containing at least t different parties, $L(Set) = h'^{sk}$ in TORTL-EC DKG for the shared secret sk .

- $L(n) == L(t)$, i.e., the interpolation that is overdefined with the evaluation of all parties is the same as the interpolation for t parties (any set of t parties can be used).
- $L(t) \neq L(t-1)$
- $L(t)$ is nontrivial

If all those checks pass, the parties now collectively possess a shared secret, which completes the protocol.

2.3. Security

We assume a broadcast channel, guaranteeing each participant receives the same message, and Peer-to-Peer (P2P) authenticated confidential communication channels between individual parties.

TORTL-ML DKG verification:

- Ensure $\|z_{i,I,1}\|_\infty$ and $\|z_{i,I,2}\|_\infty \leq \beta - \kappa \cdot n$
- Recalculate $t'_{i,I,1} = \langle b, z_{i,I,1} \rangle + chal_{i,I} \cdot eval_{i,I}$ and $t'_{i,I,2} = A \cdot z_{i,I,1} + B \cdot z_{i,I,2} + chal_{i,I} \cdot com_{i,I}$
- $chal_{i,I} = H_1(b, eval_{i,I}, com_{i,I}, A, B, t'_{i,I,1}, t'_{i,I,2})$

Finally, perform the following functional checks for each $I \in S$:

- $\forall j, j' \in I, j \neq j', eval_{j,I} == eval_{j',I}$

We model the adversary as a Probabilistic Polynomial-Time (PPT) algorithm for TORTL-EC DKG and as a Quantum Polynomial-Time (QPT) algorithm for TORTL-ML DKG, both able to perform static corruptions bounded by the duration of the established threshold setup. For a participant threshold t and a number of parties n , our schemes are secure for up to $t - 1$ static malicious corruptions, with $t \leq n$.

We operate under a "succeed or restart" mechanism, where a participant either knows that the protocol succeeded, or that it aborted, but is unable to identify the party responsible for the abort. We acknowledge that there exist numerous malicious behaviours that can cause the protocol to abort, including, but not limited to, a refusal to participate. We do not address this limitation and simply state that if the protocol succeeds, then the parties share a secret with the specified threshold t and know a public key to the shared secret, along with commitments to each individual share. This is provably enforced by our ZKPs and functional verifications. Honest parties will abort the execution if any of these verifications fail at any point.

We address in particular the threat posed by a rushing adversary. The adversary must provide a proof of knowledge of an opening of their contribution to the public key, which stops them from contributing an arbitrary value. However, it is possible for the adversary to perform a brute-force search for an opening that will force the resulting public key to satisfy some rare predicate, as long as they find such an opening in polynomial time. It is possible to limit this to common predicates by implementing a timeout before honest parties abort.

Our protocol is secure in the ROM under the DDH assumption for TORTL-EC DKG and is secure in the ROM under the MLWE and MSIS assumptions for TORTL-ML DKG. We believe TORTL-ML DKG may be adapted to be secure in the Quantum Random Oracle Model (QROM).

3. Open-Source Implementation

Public implementations of TORTL DKG code will be hosted at <https://gitlab.com/kelvin-zero/tortl>. All code will be written in Rust 1.95.0. Library crates will contain modules for model objects, cryptographic functions, and some utilities. We have external direct dependencies on `subtle`, `thiserror`, `crypto-bigint`, `shake`, and a few RustCrypto crates. All instructions to build and run will be provided in the code repository. Currently, the TORTL-EC-P256 DKG implementation is published, but the TORTL-ML DKG implementation is still in development. The definitive architecture of the networking layer is currently undergoing finalization. The final submission will include the chosen networking stack alongside scripts and configuration files for setup and testing. We plan to offer an extensive suite of Known Answer Tests (KATs), tests of the whole DKG with simulated participants, and tests with networked participants to show realistic behaviour and performance.

4. Experimental Performance Evaluation

While we are still preparing performance tests for TORTL-EC-P256 DKG and TORTL-ML DKG, we have initial results for TORTL-EC-P256 DKG (see Table 1 below). These were collected using a single process without parallelization on a single computer with 16 cores and 32 GB of RAM. We highlight that while the size of messages sent between participants is constant, the number of messages exchanged is $\Theta(n^2)$. We also observe that for constant n values, the runtime increases with t , though this impact is negligible compared to that of n , even without considering communication overhead. In a practical deployment, we expect the performance to be dominated by the network communication time. We do not expect any challenges running on the prescribed baseline platform.

Table 1: Single process runtime of TORTL-EC-P256 DKG

t	n	Runtime (ms)
2	3	21
3	4	31
4	4	35
4	8	52
8	8	61
4	16	91
8	16	99
16	16	121

t	n	Runtime (ms)
4	32	174
8	32	180
16	32	202
32	32	249
4	64	332
8	64	343
16	64	387
32	64	438
64	64	560

t	n	Runtime (ms)
4	128	678
8	128	679
16	128	742
32	128	838
64	128	950
128	128	1203

Regarding message size for TORTL-ML DKG, our current implementation uses threshold $t = 4$ and $n = 5$ and achieves a payload size of slightly less than 14.5 kB per participant per key. However, because of the redundancy inherent to RSS, each participant performs Round 2 for four different keys, resulting in a total payload size of roughly 59 kB.

5. Licensing, Patent Claims, and Funding

We disclose pending international patent application [PCT/CA2024/050389](https://patents.google.com/patent/PCT/CA2024/050389), primarily covering TORTL-EC DKG with potential applicability to TORTL-ML DKG. The core TORTL-EC-P256 DKG implementation is available under the Apache License 2.0, which explicitly grants patent rights for the contributed code; we will also provide complete licensing details for all dependencies. This research and its implementations were entirely funded by Kelvin Zero Inc., which holds the associated patent rights. The submitting team maintains the following affiliations with Kelvin Zero Inc.: Thierry St-Jacques-Gagnon (patent author) is a founder, shareholder, and employee; Nadège Pulgar-Vidal is an employee and shareholder; Maxime Godon (patent author) is a former employee, contractor, and shareholder; Florian Le Mouël (patent author) is a student.

6. Comparison with Other Submissions

6.1. TORTL-EC

SplitForge [LPSVM+26] differs from TORTL-EC DKG mainly in its trust model. SplitForge is designed specifically for use cases where one party (the server) is more secure than the other (the client), while TORTL-EC DKG is also compatible with use cases where all parties are equally responsible for security. SplitForge is also designed specifically for two-party use cases, where both parties must be secure, although not equally. TORTL-EC DKG supports an arbitrary number of parties with an arbitrary threshold.

We differ from (Red)ETA [BBLMPS26] in that we do not seek verifiable correctness of individual shares. Instead, our functional verification allows us to significantly reduce the number of commitments communicated over the network. We expect this to make our protocol faster and more appropriate for low-memory environments.

6.2. TORTL-ML

We share many similarities with Mithril's DKG [CDPENP26]: RSS, the ROM, security against dishonest majority, small-sized committees (their preview suggests $N \leq 8$), and unanimous aborts. TORTL-ML benefits from its efficient runtime since we require half as many rounds of communication as Mithril's DKG. However, Mithril does not require a broadcast channel, whereas we do.

Quorus [BCEPT26] builds on top of SSSS. Their strengths include the QROM, identifiable aborts, support for up to medium-sized quorums ($N \leq 64$), and independence from a broadcast channel. TORTL-ML benefits from our individual, offline rejection sampling and security against a dishonest majority.

SplitForge [LPSVM+26] introduces Trilithium, which includes a DKG for a three-party ML-DSA. While it is designed for small quorums, TORTL-ML allows for a wider variety of use cases, and requires only two rounds, compared to Trilithium's three. Since network communications typically dominate the execution time of a DKG, we expect TORTL-ML to be faster.

6.3. Discussion

Unlike the aforementioned schemes and other N4 candidates (e.g., Fireblock-3MI and TALUS), we submit a standalone DKG rather than a precursor to an adjoined signature scheme. Nonetheless, TORTL DKG can directly benefit submissions currently lacking a DKG, including Trout++ [DNP26], Twig [LLDZWWQ26], and KU [KUVS26]. For TORTL-ML, we assume both MLWE and MSIS, whereas other candidates typically rely only on MLWE. Compared to other DKG submissions, TORTL DKG reduces runtime and bandwidth by requiring just two communication rounds and foregoing identifiable aborts while remaining trustless.

References

- [AMMR18] Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. “DiSE: distributed symmetric-key encryption”. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 1993–2010. DOI: [10.1145/3243734.3243774](https://doi.org/10.1145/3243734.3243774). URL: <https://eprint.iacr.org/2018/727>. Also at ia.cr/2018/727.
- [BBLMPS26] Alessandro Barenghi, Michele Battagliola, Riccardo Longo, Alessio Meneghetti, Gerardo Pelosi, and Edoardo Signorini. *(Red)ETA: Refreshable Extensible DLOG Enhanced Threshold Algorithms*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/RedETA-PW01.pdf>.
- [BCEPT26] Alexander Bienstock, Leo de Castro, Daniel Escuerdo, Antigoni Polychroniadou, and Akira Takahashi. *Quorus*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Quorus-PW01.pdf>.
- [BDLOP18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. “More efficient commitments from structured lattice assumptions”. In: *International conference on security and cryptography for networks*. Springer. 2018, pp. 368–385. DOI: [10.1007/978-3-319-98113-0_20](https://doi.org/10.1007/978-3-319-98113-0_20). Also at ia.cr/2017/523.
- [CDPENP26] Sofía Celi, Gustavo Delerue, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Mithril*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Mithril-PW01.pdf>.
- [CP92] David Chaum and Torben Pryds Pedersen. “Wallet databases with observers”. In: *Annual international cryptology conference*. Springer. 1992, pp. 89–105. DOI: [10.1007/3-540-48071-4_7](https://doi.org/10.1007/3-540-48071-4_7).
- [DNP26] Hila Dahari-Garbian, Ariel Nof, and Luke Parker. *Trout++*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. May 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Trout-PW02.pdf>.
- [EKSSZSC21] Muhammed F Esgin, Veronika Kuchta, Amin Sakzad, Ron Steinfeld, Zhenfei Zhang, Shifeng Sun, and Shumo Chu. “Practical post-quantum few-time verifiable random function with applications to algorand”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2021, pp. 560–578. DOI: [10.1007/978-3-662-64331-0_29](https://doi.org/10.1007/978-3-662-64331-0_29). Also at ia.cr/2020/1222.
- [FS86] Amos Fiat and Adi Shamir. “How to prove yourself: Practical solutions to identification and signature problems”. In: *Conference on the theory and application of cryptographic techniques*. Springer. 1986, pp. 186–194. DOI: [10.1007/3-540-47721-7_12](https://doi.org/10.1007/3-540-47721-7_12).

[KUVS26] Jonathan Katz, Antoine Urban, Denis Varlakov, and Nikita Sorokovikov. *KU*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. July 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Dfns-KU-PW02.pdf>.

[LLDZWWQ26] Zengpeng Li, Yingjie Lyu, Xudong Deng, Hong-Sheng Zhou, Mei Wang, Puwen Wei, and Chen Qian. *Twig: Two-Round Threshold ECDSA*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. February 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Twig-PW02.pdf>.

[LPSVM+26] Peeter Laud, Alisa Pankova, Nikita Snetkov, Jelizaveta Vakarjuk, Petr Muzikant, Aivo Kalu, Burak Can Kus, Semjon/Sona Kravtšenko, Raul-Martin Rebane, and Mart Oruaas. *SplitKey*. Preview Writeup submitted to the NIST First Call for Multi-Party Threshold Schemes. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/SplitForge-PW01.pdf>.

[LS18] Vadim Lyubashevsky and Gregor Seiler. “Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 204–224. DOI: [10.1007/978-3-319-78381-9_8](https://doi.org/10.1007/978-3-319-78381-9_8). Also at ia.cr/2017/523.

[Lyu09] Vadim Lyubashevsky. “Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures”. In: *International conference on the theory and application of cryptology and information security*. Springer. 2009, pp. 598–616. DOI: [10.1007/978-3-642-10366-7_35](https://doi.org/10.1007/978-3-642-10366-7_35).

[Lyu12] Vadim Lyubashevsky. “Lattice signatures without trapdoors”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, pp. 738–755. DOI: [10.1007/978-3-642-29011-4_43](https://doi.org/10.1007/978-3-642-29011-4_43). Also at ia.cr/2011/537.

[MGBBBC24] Florian Le Mouël, Maxime Godon, Renaud Brien, Erwan Beurier, Nora Boulahia-Cuppens, and Frédéric Cuppens. “Trustless Distributed Symmetric-key Encryption”. In: *arXiv preprint arXiv:2408.16137* (2024). DOI: [10.48550/arXiv.2408.16137](https://doi.org/10.48550/arXiv.2408.16137).

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. “Distributed pseudo-random functions and KDCs”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 327–346. DOI: [10.1007/3-540-48910-X_23](https://doi.org/10.1007/3-540-48910-X_23).

[Ped91a] Torben Pryds Pedersen. “A threshold cryptosystem without a trusted party”. In: *Workshop on the Theory and Application of Cryptographic Techniques*. Springer. 1991, pp. 522–526. DOI: [10.1007/3-540-46416-6_47](https://doi.org/10.1007/3-540-46416-6_47).

[Ped91b] Torben Pryds Pedersen. “Non-interactive and information-theoretic secure verifiable secret sharing”. In: *Annual international cryptology conference*. Springer. 1991, pp. 129–140. DOI: [10.1007/3-540-46766-1_9](https://doi.org/10.1007/3-540-46766-1_9).

[Sha79] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613. DOI: [10.1145/359168.359176](https://doi.org/10.1145/359168.359176).

[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).