

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

Title: Kettle

Subtitle: Short Post-Quantum Threshold Signatures from the HAWK Signature Scheme

Version: Preview Writeup 1.0 (2026-09-19)¹

Team name: Kettle team

Team members: Calvin Abou Haidar, Daniel Escudero, Thomas Espitau, Clément Hoffmann, Haruhisa Kosuge, Kaoru Takemure, Mehdi Tibouchi, Hernán Darío Vanegas Madrigal

Abstract: Kettle is a threshold signature protocol that produces specification-compliant signatures for HAWK, a lattice-based hash-and-sign scheme that reached the third round of the additional call for post-quantum signatures. Threshold variants of hash-and-sign lattice signatures have so far been considered impractical, because signing requires sampling a discrete Gaussian over a secret-dependent lattice coset. We observe that HAWK avoids this obstacle: it samples each coordinate independently from a one-dimensional discrete Gaussian over the even or the odd integers, with the coset selected by secret bits. The whole signing algorithm can therefore be expressed over a small set of generic secure-computation building blocks, namely multiplication, uniform bit generation, doubly-authenticated bits and integer comparison. Instantiating those building blocks with protocols offering given security properties yields a threshold signing protocol with the same properties. We obtain a protocol that is universally composable, supports an arbitrary number of parties and thresholds, and needs as few as two online rounds and about two kilobytes of online communication per party, for signatures of about 1.2 kilobytes.

Proposed crypto-systems:

- (I) Kettle: Threshold Signing for HAWK (Categories S1)

Keywords: Threshold Cryptography; NIST Threshold Call; Threshold Signatures; Post-Quantum Cryptography; Lattice-Based Cryptography; HAWK; Hash-and-Sign; Lattice Isomorphism Problem; Secure MPC; UC

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: Calvin Abou Haidar^{i1,a1}, Daniel Escudero^{i2,a5}, Thomas Espitau^{i3,a2}, Clément Hoffmann^{i4,a1}, Haruhisa Kosuge^{i5,a1}, Kaoru Takemure^{i6,a3,a4}, Mehdi Tibouchi^{i7,a1}, Hernán Darío Vanegas Madrigal^{i8,a6}

Open Researcher and Contributor Identifiers (ORCID):

i1 ([0009-0000-0955-9140](https://orcid.org/0009-0000-0955-9140)); i2 ([0000-0003-2375-0034](https://orcid.org/0000-0003-2375-0034)); i3 ([0000-0002-7655-9594](https://orcid.org/0000-0002-7655-9594)); i4 ([0000-0001-9744-1539](https://orcid.org/0000-0001-9744-1539)); i5 ([0009-0001-9976-3344](https://orcid.org/0009-0001-9976-3344)); i6 ([0000-0002-9288-1911](https://orcid.org/0000-0002-9288-1911)); i7 ([0000-0002-2736-2963](https://orcid.org/0000-0002-2736-2963)); i8 ([0000-0002-5662-9663](https://orcid.org/0000-0002-5662-9663))

Affiliations:

^{a1} NTT Social Informatics Laboratories @ Tokyo, Japan

^{a2} PQShield @ Paris, France

^{a3} PQShield @ Tokyo, Japan

^{a4} National Institute of Advanced Industrial Science and Technology @ Tokyo, Japan

^{a5} TACEO @ Graz, Austria

^{a6} HashCloak Inc. @ Ontario, Canada

Main contacts:

- **Team mailing list:** kettle-team@googlegroups.com
- **Primary technical contact person:** Clément Hoffmann, clement.hoffmann@ntt.com
- **Secondary contact person:** Mehdi Tibouchi, mehdi.tibouchi@ntt.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

We plan to submit Kettle, a family of threshold protocols that produce specification-compliant signatures for HAWK, a lattice-based hash-and-sign signature scheme based on the LIP over rank-2 modules of power-of-two cyclotomic rings [HAWK]. HAWK reached the third round of the NIST additional call for post-quantum signatures. Its team withdrew after a cryptanalytic result reduced the conjectured hardness of the underlying module-LIP, roughly doubling the dimension needed for a given security level [HawkSVP]. While this attack affects negatively the communication cost and the signature size of our scheme, it remains relevant on both these metrics. With revised parameters, HAWK signatures are about half the size of ML-DSA signatures, signing is faster than in both ML-DSA and FALCON, and no floating-point arithmetic is involved.

Why HAWK. Threshold signing for hash-and-sign lattice signatures has so far been considered impractical. In the GPV framework used by FALCON/FN-DSA, signing samples a discrete Gaussian over a lattice coset that depends on both the message and the secret trapdoor. Reproducing that sampler on secret-shared data requires recursive lattice decompositions. On top of that, both the sampler and the other operations require high-precision arithmetic. The only concrete realization to date needs roughly a thousand online rounds per signature even after amortization [TFalcon]. HAWK avoids these difficulties by construction. It samples each coordinate independently from a one-dimensional discrete Gaussian supported on $2\mathbb{Z}$ or $2\mathbb{Z} + 1$, with the coset selected by a secret bit. Coordinate sampling, coset selection and the remaining polynomial arithmetic can all be expressed over a small set of standard MPC primitives. CDT-based sampling, in particular, simply requires comparisons, that can be round offline and in parallel. That is what makes Kettle possible, and why our submission targets HAWK rather than a NIST-specified signature: we know of no standardized signature whose signing algorithm is comparably MPC-friendly while keeping signatures this small.

Proposed crypto-systems and categories. Since HAWK is not NIST-specified, our proposal sits in Class S. We plan to submit *Kettle* (S1), a threshold signature protocol whose output is accepted by the unmodified HAWK verification algorithm.

We do not claim S4 (distributed key generation). Distributing the generation of the HAWK NTRU trapdoor would require solving the NTRU equation inside MPC, which we regard as an open problem, as does [TFalcon]. Our protocol assumes a trusted setup that securely generates the signing key and distributes shares of this secret to the parties.

Real-world relevance. Threshold signatures are a useful tool when shared authority over signing rights are necessary: shared custody of (crypto) assets, distributed certificate authorities, secure enterprise key management, etc. Kettle builds upon the HAWK signature scheme, directly adding a threshold layer to it. In particular, this means that Kettle does not introduce any new assumption at its core. Having HAWK as the underlying signature scheme also allows Kettle to benefit from its competitive signature size, as opposed to ad hoc lattice-based threshold signature protocols, which all generate much larger signatures. This makes Kettle a strong candidate in scenarios where signature size is a major constraint.

2. Specification

Organization.

1. **The ABB interface.** All protocols are described in the arithmetic black-box model, with two storage domains: an arithmetic domain over $\mathbb{Z}_q$ and a binary domain over $\mathbb{F}_2$. The commands are input, add, addmult-cons, mult, open, rand, coin and mod2, extended with daBits [daBits] and an integer comparison. This enable the submission to cover a range of threshold profiles rather than one: the security properties are inherited from whichever protocol instantiates the ABB.
2. **Preprocessing (offline signing phase).** A message-independent phase producing, per signature, a tuple consumed by the online phase. It contains the shared Gaussian sampler. It is declined in two versions: one that optimizes the number of online rounds (at the cost of offline communication scaling quadratically with the number of parties), and one that optimizes communication cost of the offline phase (scaling linearly with the number of parties, at the cost of an additional round in the online phase).
3. **Online signing phase.** We propose two variants again, corresponding to the two versions of the offline phase: a 2-round protocol (corresponding to the round-optimized offline phase) and a 3-round protocol (corresponding to the communication-optimized offline phase).

System model. The topology we consider consists of N parties with *pairwise authenticated private channels*. We do not need to assume broadcast channels. As mentioned previously, the parties are assumed to receive their shares of the signing key in a trusted setup phase (distributed key generation is left out of scope of the submission). The protocols we present focus on signature generation.

Kettle makes no restriction on the number of parties N or on the corruption threshold t . Since it is specified in the arithmetic black-box model, the chosen instantiations of the basic functionalities like multiplication determine the supported security properties and communication threshold. We have instantiated and run the protocol both in the honest-majority setting ($t < N/2$, with honest-majority Shamir secret sharing as well as the 4-party Fantastic Four protocol) and with dishonest-majority ($t \leq N - 1$, with OT-based secure computation protocol MASCOT).

Technical approach. At a high level, the HAWK signing algorithm proceeds as follows, where $\mathbf{B}$, f and g are all part of the signing key.

1. **Hashing:** Compute a public hash of the message m to be signed, a binary vector $\mathbf{h} \leftarrow H(m)$.
2. **Center Computation:** Compute the center $\mathbf{t} \leftarrow \mathbf{B} \cdot \mathbf{h} \pmod{2}$ (also a binary vector).
3. **Gaussian Sampling:** Sample a Gaussian vector $\mathbf{x}$ coordinate-wise from $\mathcal{D}^{(t_0)} \times \mathcal{D}^{(t_1)} \times \dots$, where $\mathcal{D}^{(b)}$ is a discrete Gaussian on the coset $2\mathbb{Z} + b$ centered at 0. That is, the i -th coefficient of $\mathbf{x}$ is distributed following a Gaussian supported over $2\mathbb{Z} + t_i$.
4. **Final Multiplication:** Let $\mathbf{x}$ be split into two halves $\mathbf{x} = (\mathbf{x}_0 \parallel \mathbf{x}_1)$, and interpret each $\mathbf{x}_i \in \mathbb{Z}_q^n$ as a ring element in a certain cyclotomic ring $\mathcal{R}$. Compute $w_1 \leftarrow f \cdot x_1 - g \cdot x_0$

in the ring $\mathcal{R}$. The signature is then given by $s \leftarrow (h_1 - w_1)/2$ (up to a choice of sign and a rare rejection step that do not matter for our purposes).

The Kettle protocol securely distributes these four steps roughly as follows (a more detailed exposition is provided in the next section):

1. **Hashing and center computation:** The message hash $\mathbf{h}$ is public, so the center $\mathbf{t} = \mathbf{B} \cdot \mathbf{h} \bmod 2$ can be obtained as a binary-shared secret by a local computation of the signers. This, in turn, can be converted to an arithmetic-shared value using one daBit-masked opening.
2. **Gaussian sampling:** Can be carried out by thresholdizing HAWK's RCDT sampler: the parties locally generate Boolean shares of a uniform 78-bit value, run 13 comparison circuits in parallel against the public table, and convert the resulting bits to the arithmetic domain via daBits again. Sampling both cosets and multiplexing on the secret center bit costs one secure product per coordinate.
3. **Multiplication:** The final polynomial $w = f_{x_1} - g_{x_0}$ is linear in the Gaussian vector, which lets us move every secret multiplication into preprocessing: writing $\mathbf{x} = \mathbf{s}_0 + \mathbf{c} \odot \mathbf{s}_1$, where $\mathbf{c}$ is the public masked center, the online phase of the signing procedure reduces to one opening, a public-coefficient linear combination of stored shares (which is a local computation), and one final opening.

Security. We prove the security of Kettle in the UC framework. We adopt the arithmetic black-box model, a standard MPC abstraction that captures the ability of the parties to distribute, operate on, and reconstruct secret values without committing to concrete secret-sharing instantiations or adversarial setting, e.g., passive or active adversary, honest or dishonest majority. We will propose secret sharing instantiations suitable for several settings, including both honest and dishonest majority. In our analysis, we primarily focus on the static corruption model, in which the adversary must decide which parties are compromised in advance.

Three points may be worth pointing out. First, our protocol is distribution-preserving: the signatures it outputs are exactly distributed as regular HAWK signatures, and not only accepted by HAWK verification (this is in contrast with a protocol like Mithril [Mithril] for ML-DSA).

Second, the public salt is sampled during preprocessing, hence before the message is known. It is the only preprocessed value the adversary learns early, and it means the adversary may choose the message after seeing the salt. We therefore analyze a variant signature scheme, we call *explicit-salt HAWK*, in which the signing functionality maintains a queue of unused salts so that each salt serves at most one signature, and we prove this variant is SUF-CMA. Our threshold signing functionality is justified by this security analysis of explicit-salt HAWK. It is proven by following the similar proof strategy used in the proof of explicit-salt MAYO [TMAYO; Vinaigrette-PW], though it showed the existential unforgeability, not SUF-CMA.

Third, HAWK signing restarts when the sampled Gaussian vector exceeds the verification bound. We omit that check, so Kettle can emit an invalid signature; for HAWK-1024 (now 128 bits of security) the specification bounds this probability by 2^{-121} , which is negligible and does not affect unforgeability.

As previously discussed, recent developments in AI-assisted cryptanalysis lead to an attack on HAWK that reduced its security [HawkSVP], making HAWK-1024 the 128-bit level security candidate instead of HAWK-512. For the 256-bit level, we introduce HAWK-2048 as part of the submission. The dimensions in HAWK have no influence on the inner workings of Kettle, they simply set the communication cost. Since the final Kettle signature is a HAWK signature, we inherit its sizes: HAWK-1024 gives a 1221-byte signature [HAWK].

As for othe submissions, Known-Answer Tests will be provided with the submission. Again, we benefit from building upon the HAWK architecture: verification vectors can be sourced from the HAWK specification, leaving only the fixed-randomness MPC transcripts to generate.

3. Extended Technical Overview

Naive threshold signing. The basic idea of the Kettle protocol is the following. We start with a secret-shared secret key $\llbracket \mathbf{B} \rrbracket$. Since the message m is public, the parties can locally compute $\llbracket \mathbf{t}' \rrbracket = \llbracket \mathbf{B} \rrbracket \cdot \mathbf{h}$. Now, we actually need to obtain $\llbracket \mathbf{t} \rrbracket$, where $\mathbf{t} = \mathbf{t}' \bmod 2$. If we use a power of two as the modulus, the parties can locally compute $\llbracket \mathbf{t} \rrbracket_2$ (over the binary domain) by locally reducing their shares modulo 2, but this still falls short, since we need $\llbracket \mathbf{t} \rrbracket$ (over the arithmetic domain) for the computation that follows next.

To address this we employ *daBits* [daBits], which work as follows. The parties preprocess a series of random bits $\mathbf{r} \in \{0, 1\}^{2n}$ both as $\llbracket \mathbf{r} \rrbracket_2$ (binary sharings) and $\llbracket \mathbf{r} \rrbracket$ (arithmetic sharings). The parties then compute locally and open $\mathbf{c} \leftarrow \llbracket \mathbf{t} \rrbracket_2 \oplus \llbracket \mathbf{r} \rrbracket_2$, followed by $\llbracket \mathbf{t} \rrbracket \leftarrow \mathbf{c} + \llbracket \mathbf{r} \rrbracket - 2 \cdot \mathbf{c} \odot \llbracket \mathbf{r} \rrbracket$ (where $\oplus$ is the coordinate-wise XOR operation on bits, and $\odot$ denotes the Hadamard product of integer vectors, i.e., coordinate-wise multiplication).

With $\llbracket \mathbf{t} \rrbracket$ in hand, the next step is to do the Gaussian sampling. Assume that, from a preprocessing to be specified later, the parties obtain shares of discrete Gaussian vectors $\llbracket \mathbf{x}^{(0)} \rrbracket$ and $\llbracket \mathbf{x}^{(1)} \rrbracket$ such that the coefficients of $\mathbf{x}^{(0)}$ are sampled from $\mathcal{D}^{(0)}$ (supported over $2\mathbb{Z}$) and those of $\llbracket \mathbf{x}^{(1)} \rrbracket$ from $\mathcal{D}^{(1)}$ (supported over $2\mathbb{Z} + 1$). They can multiplex these using $\mathbf{t}$ as

$$\llbracket \mathbf{x} \rrbracket = \llbracket \mathbf{x}^{(0)} \rrbracket + \llbracket \mathbf{t} \rrbracket \odot \llbracket \Delta \mathbf{x} \rrbracket,$$

where $\Delta \mathbf{x} = \mathbf{x}^{(1)} - \mathbf{x}^{(0)}$.

The above requires the parties to compute the secure product $\llbracket \mathbf{t} \rrbracket \odot \llbracket \Delta \mathbf{x} \rrbracket$, which could be done with any secure multiplication protocol. What follows then is splitting into two halves $\llbracket \mathbf{x} \rrbracket = (\llbracket \mathbf{x}_0 \rrbracket \parallel \llbracket \mathbf{x}_1 \rrbracket)$, interpreting $(\llbracket \mathbf{x}_0 \rrbracket, \llbracket \mathbf{x}_1 \rrbracket)$ as *polynomials* $(\llbracket x_0 \rrbracket, \llbracket x_1 \rrbracket)$ over $\mathcal{R}$, and computing $\llbracket w \rrbracket = \llbracket f \rrbracket \cdot \llbracket x_1 \rrbracket - \llbracket g \rrbracket \cdot \llbracket x_0 \rrbracket$. Once again, this requires secure multiplication. After this, the signature can be reconstructed with local computation only.

Optimizing the naive baseline. The naive online phase has two costly secure-multiplication steps: the Hadamard product $\llbracket \mathbf{t} \rrbracket \odot \llbracket \Delta \mathbf{x} \rrbracket$ and the polynomial product $\llbracket f \rrbracket \cdot \llbracket x_1 \rrbracket - \llbracket g \rrbracket \cdot \llbracket x_0 \rrbracket$ over $\mathcal{R}$. Neither of them genuinely depends on the message: the only message-dependent quantity in the whole online phase is the public binary vector $\mathbf{c}$ produced by the daBit opening. The

optimization below moves *all* secret multiplications into the preprocessing, leaving the online phase with two openings and a single public-coefficient linear combination of stored shares.

Linearizing the final polynomial. Let $\text{rot}(h) \in \mathbb{Z}_q^{n \times n}$ denote the matrix of multiplication by $h \in \mathcal{R}$ in the coefficient basis. Then the HAWK final polynomial $w = f \cdot x_1 - g \cdot x_0$ has coefficient vector $\mathbf{w} = \text{rot}(f) \cdot \mathbf{x}_1 - \text{rot}(g) \cdot \mathbf{x}_0$, which we write compactly as $\mathbf{w} = \mathbf{M} \cdot \mathbf{x}$ with $\mathbf{x} = (\mathbf{x}_0 \parallel \mathbf{x}_1)$ and $\mathbf{M} \stackrel{\text{def}}{=} [-\text{rot}(g) \mid \text{rot}(f)] \in \mathbb{Z}_q^{n \times 2n}$. Since each entry of $\mathbf{M}$ is a signed copy of one coefficient of f or g , we have $\llbracket \mathbf{M} \rrbracket$ “for free” from $\llbracket f \rrbracket, \llbracket g \rrbracket$.

Absorbing the multiplex into the mask. Substituting the daBit identity $\mathbf{t} = \mathbf{c} + (\mathbf{1} - 2\mathbf{c}) \odot \mathbf{r}$ into the multiplex $\mathbf{x} = \mathbf{x}^{(0)} + \mathbf{t} \odot \Delta \mathbf{x}$ and rearranging, we have

$$\mathbf{x} = \underbrace{\mathbf{x}^{(0)} + \mathbf{r} \odot \Delta \mathbf{x}}_{\stackrel{\text{def}}{=} \mathbf{s}_0} + \mathbf{c} \odot \underbrace{(\mathbf{1} - 2\mathbf{r}) \odot \Delta \mathbf{x}}_{\stackrel{\text{def}}{=} \mathbf{s}_1}.$$

The split $\mathbf{x} = \mathbf{s}_0 + \mathbf{c} \odot \mathbf{s}_1$ separates the secret preprocessed randomness ($\mathbf{s}_0, \mathbf{s}_1$, derived from $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{r}$) from the public binary coefficients ($\mathbf{c}$, opened online).

Packaging the preprocessing. Composing the two steps,

$$\mathbf{w} = \mathbf{M} \cdot \mathbf{x} = \underbrace{\mathbf{M} \cdot \mathbf{s}_0}_{\stackrel{\text{def}}{=} \mathbf{w}_0} + \mathbf{M} \cdot (\mathbf{c} \odot \mathbf{s}_1).$$

The first summand $\mathbf{w}_0$ is fully determined by the preprocessing. The second summand depends on $\mathbf{c}$ but only *linearly*: expanding $(\mathbf{c} \odot \mathbf{s}_1)_i = c_i \cdot (\mathbf{s}_1)_i$, its j -th coordinate is

$$(\mathbf{M} \cdot (\mathbf{c} \odot \mathbf{s}_1))_j = \sum_{i=1}^{2n} \mathbf{M}_{j,i} \cdot (\mathbf{s}_1)_i \cdot c_i = (\mathbf{V} \cdot \mathbf{c})_j,$$

where $\mathbf{V}_{j,i} \stackrel{\text{def}}{=} \mathbf{M}_{j,i} \cdot (\mathbf{s}_1)_i$. Both $\mathbf{w}_0$ and $\mathbf{V}$ are products of secret-key shares and preprocessed randomness; given them, the entire variable part of $\mathbf{w}$ is the linear combination $\mathbf{V} \cdot \mathbf{c}$ with the public coefficients $\mathbf{c}$, computed locally on stored shares.

The optimized online phase. Assuming the preprocessing has stored $(\llbracket \mathbf{w}_0 \rrbracket, \llbracket \mathbf{V} \rrbracket, \llbracket \mathbf{r} \rrbracket_2)$, signing reduces to (i) one opening to publish $\mathbf{c} \leftarrow \text{open}(\llbracket \mathbf{t} \rrbracket_2 \oplus \llbracket \mathbf{r} \rrbracket_2) \in \mathbb{F}_2^{2n}$; (ii) the local computation $\llbracket \mathbf{w} \rrbracket \leftarrow \llbracket \mathbf{w}_0 \rrbracket + \llbracket \mathbf{V} \rrbracket \cdot \mathbf{c}$, using only public coefficients $\mathbf{c}$ and stored shares; and (iii) one opening to recover $\mathbf{w}$, and follow the final *public* steps of HAWK signing. No secret multiplication takes place online. The preprocessing itself grows quadratically in n (it computes $\Theta(n^2)$ secret products to assemble $\llbracket \mathbf{w}_0 \rrbracket$ and $\llbracket \mathbf{V} \rrbracket$); we also propose an alternative scheme with linear-communication preprocessing, at the cost of one extra online round.

Preprocessing the Gaussians. In the above we assumed that the parties can sample shared Gaussian vectors $\llbracket \mathbf{x}^{(0)} \rrbracket$ from $\mathcal{D}^{(0)}$ (supported over $2\mathbb{Z}$) and $\llbracket \mathbf{x}^{(1)} \rrbracket$ from $\mathcal{D}^{(1)}$ (supported over $2\mathbb{Z} + 1$). To do this, we directly adapt the discrete Gaussian sampling algorithm from HAWK itself to the multi-party setting.

More precisely, discrete Gaussian sampling in HAWK is carried out by so-called reverse cumulative distribution table (RCDT) sampling. For each of the two distributions $\mathcal{D}^{(b)}$, $b = 0, 1$, one tabulates

the probabilities $p_i^{(b)} = \Pr_{x \sim \mathcal{D}^{(b)}}[x \geq i]$ with sufficiently many bits of precision L (in the case of HAWK, it is shown that $L = 78$ suffices): these are the RCDTs. Then, one can sample from $\mathcal{D}^{(b)}$ by sampling a random integer u of L bits, comparing it to every value $p_i^{(b)}$ in the corresponding RCDT, and counting the number of indices i such that $u \leq p_i^{(b)}$.

In Kettle, we proceed similarly: to generate a shared Gaussian sample from $\mathcal{D}^{(0)}$, say, the parties first sample Boolean shares $\llbracket u_i \rrbracket_2$ of the bits of the L -bit value u (by locally generating uniform bits), and apply a Boolean comparison circuit to these bits to obtain the shared bit $\llbracket b_i \rrbracket_2 = \llbracket u \leq p_i^{(0)} \rrbracket_2$ for each i (i.e., $b_i = 1$ if and only if $u \leq p_i^{(0)}$). By using a comparison circuit of logarithmic depth and linear size such as the one arising from the Brent–Kung adder circuit [BK82], this can typically be done in roughly $\log_2 L$ rounds and a total of $2L$ multi-party bit multiplications (i.e., multi-party AND gates). The number of indices i such that $u \leq p_i^{(0)}$, from which the Gaussian sample is easily deduced, can then be obtained by converting these Boolean shares to arithmetic shares $\llbracket b_i \rrbracket$ using daBit openings, and summing them together (which is a local computation).

4. Implementation, Performance, Licensing and Patent Claims

1. **Code progress and availability.** As for now, we only provide a proof-of-concept MP-SPDZ program, `kettle.mpc`, written in a Python-like description language, together with an analytic model of per-party communication, `costs.py`.

As part of this submission, we intend to provide an optimized implementation of Kettle, targeting the several parameter sets.

Key generation is not yet implemented, the underlying keygen of HAWK will be used and, assuming a trusted setup is available, HAWK keys will be shared accordingly. The code is currently public in an anonymized repository at <https://anonymous.4open.science/r/kettle-85DE/>, because of a concurrent academic submission.

2. **Testing.** Correctness can be checked using reference HAWK signatures, since our protocol emits exact signatures. We are also able to evaluate the correctness of our Gaussian sampler against test vectors from the HAWK reference sampler.
3. **Efficiency.** For $N = 1023$ signers in the honest majority setting (using honest-majority Shamir secret sharing, and the online round-optimized version of our protocol), final signature size is 1221 B, for 2 online rounds and 1.9 kB online communication per party, and 10 offline rounds with 36.6 MB offline communication per party.
4. **Licensing.** The code of our submission will be open source, licensed under the Apache License, Version 2.0.
5. **Patent claims.** None of the co-author have any patent related to this submission, and we are not aware of any patent that could apply to Kettle.

5. Comparison with Other Submissions

Kettle is built by thresholdizing a pre-existing lattice signature, as opposed to purpose-built threshold schemes. Here, we give comparisons with similar schemes.

Versus threshold ML-DSA. The Quorus [Quorus; Quorus-PW] and Mithril [Mithril; Mithril-PW] protocols thresholdize the ML-DSA Fiat–Shamir with aborts signature, and incur significant overhead from rejected signatures, as well as from the nonlinear parts in signing generation (such as bit extraction and norm checks). Quorus is modular and scales to an arbitrary number of parties, but requires 16 online rounds not including restarts. Mithril needs 3 rounds per attempt but is restricted to at most six parties, and its output distribution is distinguishable from that of standard ML-DSA. TALUS [TALUS-PW] is a further threshold ML-DSA preview in the Call, which has only 2 online rounds even after rejection sampling and supports arbitrary threshold, but its offline round complexity is superconstant (scaling logarithmically with the number of parties) and it requires an honest majority.

In contrast, Kettle has no rejection sampling, constant round complexity online and offline (with 2 online rounds, or 3 in the communication-optimized setting), scales to any number of parties, is distribution-preserving, and can be instantiated with a dishonest majority. Its signatures are roughly half the size of the ML-DSA schemes at the 128-bit security level: 1221 B for HAWK-1024 vs. 2560 B for ML-DSA-44.

Versus threshold FALCON. Threshold FALCON [TFalcon] is the closest relative. It is also a thresholdization of a lattice-based hash-and-sign, relying on Gaussian sampling and producing short signatures. However, its cost makes it impractical, with about a thousand amortized online rounds after compared to 2 or 3 for Kettle without amortization.

Versus purpose-built threshold lattice signatures. Other lattice-based threshold signatures including Pelican [Pelican], TRaccoon [TRaccoon], EKT [EKT], Ringtail [Ringtail] and Olingo [Olingo] are design from the ground-up as threshold protocols instead of thresholdizing a pre-existing signature (or in the case of Raccon, the thresholdized scheme itself was designed as threshold-friendly). They generally have a structure similar to the Fiat–Shamir with aborts scheme, but replace rejection with noise flooding, and hence eliminate the overhead of rejection sampling altogether. This enables them to achieve very low round complexity, in some cases a single online round, which beats Kettle. Accommodating for noise flooding, however, requires a much larger modulus and results a significant increase in signature size, even when applying some form of signature compression. Due to this, these schemes produce signatures roughly 10 to 20 times larger than Kettle’s. Hermine [Hermine-PW] and Tanuki [Tanuki-PW], two Raccoon-based previews in the Call, belong to this family and share its pros and cons.

References

[BK82] Richard P. Brent and H. T. Kung. “A Regular Layout for Parallel Adders”. In: *IEEE Transactions on Computers* C-31.3 (March 1982), pp. 260–264. DOI: [10.1109/TC.1982.1675982](https://doi.org/10.1109/TC.1982.1675982). Also available from [the author’s website](#).

[daBits] Dragos Rotaru and Tim Wood. “MARbled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security”. In: *Progress in Cryptology — INDOCRYPT 2019*. Vol. 11898. Lecture Notes in Computer Science. Springer, 2019, pp. 227–249. DOI: [10.1007/978-3-030-35423-7_12](https://doi.org/10.1007/978-3-030-35423-7_12). Also at ia.cr/2019/207.

[EKT] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. “Two-Round Threshold Signature from Algebraic One-More Learning with Errors”. In: *Advances in Cryptology — CRYPTO 2024, Part VII*. Vol. 14926. Lecture Notes in Computer Science. Springer, 2024, pp. 387–424. DOI: [10.1007/978-3-031-68394-7_13](https://doi.org/10.1007/978-3-031-68394-7_13). Also at ia.cr/2024/496.

[HAWK] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. *HAWK*. Tech. rep. National Institute of Standards and Technology, 2024. URL: <https://hawk-sign.info>.

[HawkSVP] Zygimantas Straznickas and Stephen A. Weis. *HAWK-n Key Recovery Reduces to SVP in Dimension $n/2 + 1$* . Tech. rep. Anthropic, July 2026. URL: https://www-cdn.anthropic.com/e8d50c167ad47beeb03d6109a4a484be95cb38ea/hawk_key_recovery.pdf.

[Hermine-PW] Giacomo Borin, Sofía Celi, Rafaël del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. *Hermine: An Efficient Raccoon-Style Non-Interactive Threshold Signature with Advanced Properties*. Preview writeup (version 1.0) for the NIST Threshold Call. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Hermine-PW01.pdf>.

[Mithril] Sofía Celi, Rafael del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. “Efficient Threshold ML-DSA”. In: *USENIX Security 2026: 35th USENIX Security Symposium*. Referred to as “Mithril” by the authors’ NIST Threshold Call preview writeup. USENIX Association, 2026. Also at ia.cr/2026/013.

[Mithril-PW] Sofía Celi, Gustavo Delerue, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *Mithril: Efficient Threshold ML-DSA from Secret Sharing with Short Shares*. Preview writeup (version 1.0) for the NIST Threshold Call. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Mithril-PW01.pdf>.

[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).

[Olingo] Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. *Olingo: Threshold Lattice Signatures with DKG and Identifiable Abort*. Cryptology ePrint Archive, Report 2025/1789. 2025. URL: <https://eprint.iacr.org/2025/1789>.

[Pelican] Thomas Espitau, Guilhem Niot, and Thomas Prest. “Flood and Submerge: Distributed Key Generation and Robust Threshold Signature from Lattices”. In: *Advances in Cryptology — CRYPTO 2024, Part VII*. Vol. 14926. Lecture Notes in Computer Science. Springer, 2024, pp. 425–458. DOI: [10.1007/978-3-031-68394-7_14](https://doi.org/10.1007/978-3-031-68394-7_14). Also at ia.cr/2024/959.

[Quorus] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. “Quorus: Efficient, Scalable Threshold ML-DSA Signatures from MPC”. In: *USENIX Security 2026: 35th USENIX Security Symposium*. USENIX Association, 2026. Also at ia.cr/2025/1163.

[Quorus-PW] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. *Quorus: Scalable Threshold ML-DSA from MPC*. Preview writeup (version 0.1) for the NIST Threshold Call. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Quorus-PW01.pdf>.

[Ringtail] Cecilia Boschini, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Akira Takahashi, and Mehdi Tibouchi. “Ringtail: Practical Two-Round Threshold Signatures from Learning with Errors”. In: *2025 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 2025, pp. 149–164. DOI: [10.1109/SP61157.2025.00070](https://doi.org/10.1109/SP61157.2025.00070). Also at ia.cr/2024/1113.

[TALUS-PW] Leo Kao and Raymond Chang. *TALUS: FIPS-204-Exact Threshold ML-DSA via Boundary Clearance*. Preview writeup (version 0.22) for the NIST Threshold Call. August 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/TALUS-PW02.pdf>.

[Tanuki-PW] Cecilia Boschini, Thomas Espitau, Aaron Kaiser, Shuichi Katsumata, Darya Kaviani, Russell W. F. Lai, Giulio Malavolta, Thomas Prest, Peter Schwabe, Akira Takahashi, Kaoru Takemure, and Mehdi Tibouchi. *Tanuki: Two-round Threshold Signatures from Lattices*. Preview writeup (version 0.1) for the NIST Threshold Call. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Tanuki-PW01.pdf>.

[TFalcon] Radhika Garg, Daniel Escudero, Antigoni Polychroniadou, Akira Takahashi, and Xiao Wang. “Thresholdizing Standardized FALCON Signatures”. In: *ACM CCS 2026: 33rd Conference on Computer and Communications Security*. ACM Press, 2026. Also at ia.cr/2026/1300.

[TMAYO] Diego F. Aranha, Giacomo Borin, Sofía Celi, and Guilhem Niot. “Optimizing and Implementing Threshold MAYO”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2026.3 (2026), pp. 489–517. DOI: [10.46586/tches.v2026.i3.489-517](https://doi.org/10.46586/tches.v2026.i3.489-517). Also at ia.cr/2026/710.

[TRaccoon] Rafaël del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. “Unmasking TRaccoon: A Lattice-Based Threshold Signature with an Efficient Identifiable Abort Protocol”.

In: *Advances in Cryptology — CRYPTO 2025, Part VI*. Lecture Notes in Computer Science. Springer, 2025, pp. 423–456. DOI: [10.1007/978-3-032-01887-8_14](https://doi.org/10.1007/978-3-032-01887-8_14). Also at ia.cr/2025/849.

[Vinaigrette-PW] Diego F. Aranha, Ward Beullens, Giacomo Borin, Fabio Campos, Sofia Celi, Basil Hess, Matthias J. Kannwischer, Lisa Kohl, and Guilhem Niot. *Vinaigrette: An MPC Approach for Threshold UOV and MAYO*. Preview writeup (version 1.0) for the NIST Threshold Call. January 2026. URL: <https://csrc.nist.gov/csrc/media/Projects/threshold-cryptography/documents/TCall-1/Vinaigrette-PW01.pdf>.