Thresholdizing Standardized FALCON Signatures

Radhika Garg* Daniel Escudero† Antigoni Polychroniadou‡ Akira Takahashi§
Xiao Wang¶

Abstract

Threshold signatures allow a quorum of parties to jointly produce a signature while preventing any smaller subset from doing so. Following NIST's post-quantum standardization, designing threshold schemes compatible with the newly selected primitives is a pressing task. In particular, no prior threshold signature scheme produces signatures verifiable under the **unmodified** FALCON verification algorithm — the NIST-selected post-quantum scheme with the smallest signatures and keys.

In this work, we present the first such threshold FALCON signing protocol, establishing its feasibility. Our technical contributions are threefold. First, we adapt the MPC-based discrete Gaussian sampling protocol of Wei et al. [WYF+23] to support *private* centers and standard deviations, as required by FALCON's signing process. Second, we carry out a Rényi divergence analysis of the Klein sampler under fixed-point arithmetic, showing that 73 bits of precision suffice to achieve the same security as the FALCON specification. Third, we design an efficient MPC protocol for the Klein sampler that exploits the fixed trapdoor basis to construct a pseudorandom correlation generator for authenticated VOLE using only two-party DPFs, reducing per-signature communication significantly over standard authenticated triple generation. We implement and benchmark our protocol in two settings: N-party signing with all-but-one corruption, and 3-party signing with honest majority, demonstrating that threshold FALCON signing is feasible for applications where compatibility with the FALCON standard is required.

1 Introduction

Threshold signatures enable a set of parties holding secret sharing of the signing key to produce a signature without revealing any information about the signing key. It requires a large enough quorum of participants, preventing smaller sets of participants from signing any message. This cryptographic tool has been widely studied and has many applications, including distributed versions of cryptocurrency wallets, certificate authorities, and credential issuance. In particular, the imminent threat posed by quantum computers has motivated the search for *post-quantum* threshold signatures. Among all post-quantum signature schemes, ML-DSA [Pub24] and FALCON [PFH+20a] have been selected by NIST for standardization, and their real-world deployment has already started. While there is rapid progress in threshold post-quantum signatures, especially for lattice-based schemes (see Section 1.2), almost all of them propose new signature schemes that are more friendly for thresholdization (than NIST-approved schemes). This is a crucial limitation from a deployment perspective: industry practitioners have already started the effort

*Northwestern University, radhikaradhika2028@u.northwestern.edu
†TACEO, daniel.escudero@protonmail.com
‡J.P.Morgan AI Research & AlgoCRYPT CoE, antigonipoly@gmail.com
§J.P.Morgan AI Research & AlgoCRYPT CoE, takahashi.akira.58s@gmail.com
¶Northwestern University, wangxiao@northwestern.edu

to deploy post-quantum cryptography, and most production cryptographic libraries (e.g., liboqs) only support NIST-selected signature schemes. Although several recent (and concurrent) works [DKLS25, BdCE+26, CdPE+26, RRU26, PPS26] have explored thresholdization of ML-DSA, the other NIST-selected lattice-based scheme, FALCON, has not yet been studied in this context. This is a particularly important open question given that FALCON offers significantly smaller signatures and verification keys than ML-DSA, which is a crucial metric in many applications.

1.1 Our Contribution

In this work, we present the *first* threshold version of the FALCON signature scheme, where the produced signature can be verified by the *unmodified* FALCON verification algorithm—a property that, as argued above, is crucial for most of the applications of threshold signing. Our signing protocol is *MPC-agnostic*: it is described in the arithmetic black-box (ABB) model and can be instantiated with any MPC protocol realizing the ABB functionality, inheriting its security guarantees (including the corruption threshold, abort behavior, and adversary type). In more detail:

- **FALCON made friendlier to MPC.** At the center of the signing process is the high-dimensional Gaussian sampling, which was instantiated by fast Fourier orthogonalization (FFO) [DP15] in the FALCON specification. The FFO-based sampling requires extensive computation on double-precision floating-point numbers, and we observed that the Klein sampler [Kle00] has greater potential in the MPC setting. To this end, we propose an efficient one-dimensional Gaussian sampling with *private* center and variance. With this optimization, we are able to perform signing only with secure integer operations, thus more friendly to MPC. Notably, our MPC Gaussian sampling technique may be of independent interest beyond this application. As a result, Falcon-512 requires 1.83B AND gates, while our sampler requires 140M AND gates and 120K daBits per signature.

- **Rényi divergence analysis of fixed-point Klein sampling.** FALCON's original precision analysis does not account for the specific algorithms used in the one-dimensional Gaussian sampling step. Since we propose a new MPC-friendly one-dimensional Gaussian sampling scheme, we provide a corresponding Rényi divergence analysis under fixed-point arithmetic, showing that 73 bits of fixed-point precision suffice to achieve the same security level as the FALCON spec.[1]

- **Efficient instantiations for full-threshold and active security.** Once the signing process is made MPC-friendly, it can be connected to any MPC protocol. Here, we focus on the most demanding instantiation: N signers, out of which up to $N-1$ may be corrupted by a static, active adversary. We observe that in this case, while other components can be instantiated using general-purpose MPC [DPSZ12, KOS16] and DaBits [RW19], the bottleneck is the repeated vector inner product in the Klein sampler. To improve its efficiency, we design a new correlation (namely, authenticated VOLE) and an efficient pseudorandom correlation generator to produce it with sublinear communication. Our crucial observation is that one of the operands of the inner product does not change across multiple signatures; thus, the correlation can be generated using lightweight building blocks. As a result, this reduces the communication cost of each inner product for Falcon-512 by a factor of $10M$ for $N = 4$ parties.

- **Implementation and Benchmarking.** We implement and benchmark our scheme to assess the feasibility of threshold FALCON signing. Even though other hash-and-sign schemes that are

[1] As a concurrent and independent work, De Almeida Braga, Fouque, Lachguel, and Prest [BFLP26] perform a new precision analysis based on absolute error bounds, and propose a fixed-point implementation of FALCON.

specifically tailored for the threshold setting such as [ENP24] offer significant efficiency gains, our results show that in applications where NIST-selected FALCON is the only choice, a threshold version is not unfeasible.

We note that key generation in our protocol ($\mathsf{NTRUGen}$ and key distribution) is modeled as a trusted setup within $\mathcal{F}_{\mathsf{ThFALCON}}$. Designing a better-than-generic distributed key generation protocol for FALCON — which would require solving the NTRU equation in MPC — is an orthogonal open problem that we leave for future work.

1.2 Related Work

FALCON is an instantiation of the GPV hash-and-sign paradigm [GPV08] over NTRU lattices [HPS98, HHP+03]. In the literature, only a few works have explored threshold signatures based on the hash-and-sign paradigm over lattices. Bendlin-Krehbiel-Peikert [BKP13] present a threshold GPV-style scheme by instantiating an information-theoretic MPC protocol for the Micciancio-Peikert (MP12) trapdoor sampling algorithm [MP12]. Albrecht-Lai-Lapiha-Woo [ALLW25] propose an algebraic approach to distributing MP12 lattice trapdoors, which can be used to thresholdize GPV more efficiently than [BKP13]. Espitau-Niot-Prest [ENP24] propose an efficient threshold hash-and-sign scheme based on the Plover signature [EEN+24]. However, none of these approaches are compatible with FALCON due to their entirely different structures in the preimage sampling algorithm.

Other than the hash-and-sign paradigm, there is rapid progress in threshold lattice-based signatures based on the Fiat-Shamir paradigm. The majority of them propose specialized MPC protocols for secret-shared signing keys [DOTT22, ADP24, CATZ24, dENP25, PN25, BCd+25, TPCZ24, DKLS25, BdCE+25, DKM+24, EKT24, KRT24, BKL+25, PKN+25, BCdP+26, GTJ+26, JTWZ26, JWZ26], while other works rely on threshold homomorphic encryption [BGG+18, ASY22, GKS24, GHK+25].

One of the main building blocks of our protocol is MPC for one-dimensional discrete Gaussian sampling, which has been studied by Cannone et al. [CKS20], and subsequent works in the context of distributed noise generation for differential privacy (DP) [WYF+23, FW24] and FHE keys [LNK+25]. In this work, we build on the approach of Wei et al. [WYF+23], who proposed an efficient rejection sampling-based discrete Gaussian sampling protocol for MPC. Our approach extends theirs to support *private* fixed-point centers and standard deviations, making it usable in threshold FALCON and fine-tuning it to the required parameters.

2 Technical Overview

Since FALCON follows the GPV hash-and-sign paradigm [GPV08], signing uses a public matrix $\mathsf{pk} = \mathbf{A}$ and a short trapdoor basis $\mathsf{sk} = \mathbf{B}$: given a message m, the signer samples a random salt r, hashes (r, m) to a target $\mathbf{t}$ and uses $\mathbf{B}$ to find a short preimage $\mathbf{s}$ with $\mathbf{As} = \mathbf{t} \bmod q$ whose distribution leaks nothing about $\mathbf{B}$; the signature is $(\mathbf{s}, r)$.

FALCON instantiates GPV over NTRU lattices [HPS98] in the ring $R_q = \mathbb{Z}[x]/(x^n + 1, q)$, with secret basis $\mathbf{B} = [[\bar{g}, \bar{G}], [-\bar{f}, -\bar{F}]] \in R^{2\times 2}$ satisfying the NTRU equation $\bar{f}\bar{G} - \bar{g}\bar{F} \equiv q \bmod (x^n + 1)$, and public key $h = \bar{g}\bar{f}^{-1} \bmod q$. Assuming parties hold secret shares $[\![\mathbf{B}]\!]$, threshold signing proceeds as:

1. Upon receiving m, parties agree on a random salt r using a secure coin tossing protocol, and locally compute a hash $t = H(m, r) \in R_q$.

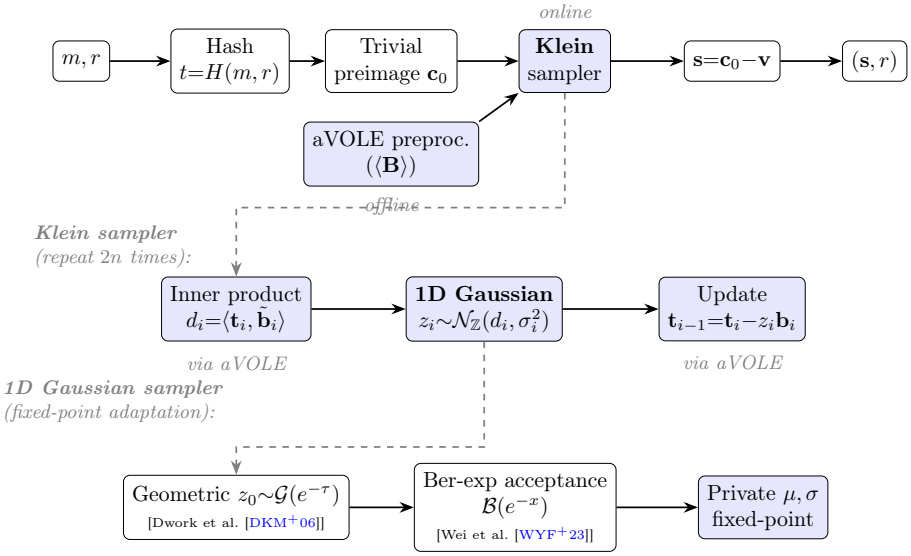


Figure 1: **Overview of our threshold FALCON signing protocol.** Shaded (blue) boxes are our contributions; unshaded boxes are standard or from prior work. The *online* phase runs the Klein sampler using aVOLE correlations generated *offline* from the shared trapdoor $\langle\mathbf{B}\rangle$. The Klein sampler iterates $2n$ times; each iteration calls our 1D Gaussian sampler (row 3), which builds on the geometric sampler of Dwork et al. [DKM+06] and the Ber-exp technique of Wei et al. [WYF+23], and contributes support for *private* center μ and std σ under fixed-point arithmetic. Our Rényi divergence analysis (Section 5) shows that 73 bits of fixed-point precision suffice for the full Klein sampler.

2. Parties locally define a trivial (not necessarily short) preimage $\mathbf{c}_0$ satisfying $[1\ h] \cdot \mathbf{c}_0 = t \bmod q$, such as $\mathbf{c}_0 = (t, 0)$.

3. Using the trapdoor basis $[\![\mathbf{B}]\!]$, parties run a *multi-party Gaussian sampling protocol* to obtain a vector $[\![\mathbf{v}]\!]$, where $\mathbf{v} \in \Lambda$ follows discrete Gaussian centered at $\mathbf{c}_0$.

4. Parties reconstruct $\mathbf{v}$, locally compute $\mathbf{s} = \mathbf{c}_0 - \mathbf{v}$, and output the signature $(\mathbf{s}, r)$. Since $[1\ h] \cdot \mathbf{v} = \mathbf{0} \bmod q$ and $\mathbf{v}$ follows a discrete Gaussian centered at $\mathbf{c}_0$, the difference $\mathbf{s}$ is indeed a preimage of t and has small norm with high probability.

2.1 MPC-friendly Gaussian sampling

The bottleneck is the multi-party Gaussian sampling step (Step 3); all other steps are straightforward in MPC. We instantiate it via the Klein sampler [Kle00], which reduces high-dimensional Gaussian sampling to iterated 1D Gaussians and linear algebra, without changing verification or parameters.

Why Klein, not FFO? The FALCON specification uses the fast Fourier orthogonalization (FFO) sampler [DP15], which decomposes the high-dimensional Gaussian sampling into a recursive tree of one-dimensional Gaussians. FFO offers *no round-complexity advantage* over Klein, and requires double-precision floating-point arithmetic. First, although the recursion tree has $O(\log n)$ levels, at each node the two recursive sub-calls are sequential (the inputs to the second depend on the output of the first), so the two cannot be parallelized. Since this dependency repeats at every node along the critical path, the FFO sampler has the same $O(n)$ sequential round complexity as the Klein sampler. Second, FFO relies heavily on double-precision floating-point arithmetic for its security, and there is currently no fixed-point precision analysis for it, making it inefficient for MPC.

MPC-friendly 1D Gaussian sampling via geometric distribution. Our first technical contribution is to design an MPC-friendly sampler that produces samples from $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$, a one-dimensional discrete Gaussian with center μ and variance σ^2, both of which are *private*. FALCON's `SamplerZ` accomplishes this via rejection sampling: it draws z_0 from the half-Gaussian $\mathcal{N}_{\mathbb{Z}^+}(0, \sigma^2_{\max})$ using inverse transform sampling, then accepts with probability $\propto e^{-x}$ where x depends on z_0, μ, and σ. Implemented naively in MPC, this requires 32 secure comparisons between high-precision numbers (for the inverse-transform step; see Section 5 for the M=32 tail cut-off analysis) and double-precision arithmetic (for the acceptance probability e^{-x}).

We reduce both costs. Instead of sampling z_0 from the half-Gaussian via inverse transform sampling, we sample from a *geometric distribution* $\mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$ with parameter τ and fold the correction into the acceptance probability. This is justified by standard rejection sampling: the geometric distribution (which decays as $e^{-\tau z_0}$) serves as an envelope for the half-Gaussian (which decays as $e^{-z_0^2/(2\sigma^2_{\max})}$), and the acceptance probability corrects for the shape difference. Combining this with the original acceptance step of `SamplerZ`, our modified sampler proceeds as follows, with changes from `SamplerZ` highlighted in red:

1. Sample geometric $z_0 \sim \mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$, where $\tau = 0.5$.
2. Transform z_0 into $z = (2b - 1)z_0 + b$, where $b \in \{0, 1\}$ is a uniform random bit.
3. With probability $p \propto \exp\left(-\frac{(z_0 - \sigma^2_{\max} \cdot \tau)^2}{2\sigma^2_{\max}} - \frac{(z-\mu)^2}{2\sigma^2} + \frac{z_0^2}{2\sigma^2_{\max}}\right)$, output z; otherwise, repeat from Step 1.

In Theorem 1, we prove that such z_0 conditioned on acceptance follows $\mathcal{N}_{\mathbb{Z}^+}(0, \sigma^2_{\max})$, so the output follows $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$ as desired.

The geometric distribution yields two further advantages in MPC. First, by the bit-decomposition property of [DKM+06], the i-th bit of a sample from $\mathcal{G}_{\mathbb{Z}^+}(p)$ follows an independent Bernoulli distribution $\mathcal{B}(p^{2^i}/(1+p^{2^i}))$. Since these probabilities are *public*, each bit can be sampled with a single secure comparison against a public constant, requiring only 5 comparisons in total (and no additions) for our parameters. In contrast, an inverse-transform approach requires 32 comparisons between high-precision numbers and 31 additions — making our geometric-based approach substantially more efficient. Second, this bitwise sampling yields z_0 directly in *Boolean shares*, which feeds naturally into the acceptance step without any share conversion: writing the exponent as $x = \sum_i 2^i x_i$ in fixed-point, we have $e^{-x} = \prod_i e^{-2^i x_i}$, so $\mathcal{B}(e^{-x})$ can be sampled by independently drawing $\mathcal{B}(e^{-2^i})$ for each bit position where $x_i = 1$ and taking the AND of all results [WYF+23]. The entire Gaussian sampling procedure thus operates on Boolean shares.

Although the geometric-based approach introduces an additional rejection step for the half-Gaussian (increasing the expected number of outer iterations), the per-iteration cost is lower. Accounting for the full cost of both the base sampler and the acceptance probability computation across all iterations, the total gate count is approximately 28K gates per sample compared to 34K for the original `SamplerZ` approach.

Rényi divergence analysis of fixed-point Klein sampling. Since our sampler operates on fixed-point representations of μ, σ, and the acceptance exponent, we must verify that the resulting distribution remains close enough to the ideal discrete Gaussian to preserve FALCON's security guarantees. FALCON's original precision analysis [HPRR20] does not account for the specific one-dimensional sampling subroutine we introduce, so we carry out a new analysis. In Section 5, we bound the Rényi divergence between the output distribution of our fixed-point sampler and the ideal discrete Gaussian as a function of the fixed-point precision f, then compose this across

all $2n$ dimensions of the Klein sampler to bound the total divergence over Q_s signing queries. We conclude that $f = 73$ bits of fixed-point precision suffice to achieve the same bit security as the latest FALCON analysis [FGdG+24], with up to $Q_s = 2^{64}$ signing queries.

2.2 Customized Cryptographic Protocols for Klein Sampler

Given the above MPC-friendly 1D Gaussian sampler, the rest of the Klein sampler works like the Gram–Schmidt process: we repeatedly project the target vector onto a normalized basis derived from the trapdoor $\mathbf{B} \in \mathbb{Z}^{2n \times 2n}$. In the i-th iteration, we compute the inner product of the current target vector $\mathbf{t}_i$ with a basis vector $\tilde{\mathbf{b}}_i$, where $\mathbf{t}_i$ depends on the outputs of all previous iterations. This step is very heavy as it requires $O(n)$ vector inner products of length $2n$ per signature, corresponding to $O(n^2)$ secure multiplications.

In the dishonest-majority setting with an active adversary, these multiplications would typically require authenticated (SPDZ-style) multiplication triples. State-of-the-art PCG-based protocols can generate such triples with sublinear communication, but authenticated triples are degree-three relations, requiring three-party distributed point functions (DPFs) — which are substantially less efficient than two-party DPFs.

Our key observation is that the same trapdoor basis $\mathbf{B}$ is reused across all signatures, so the basis vectors $\tilde{\mathbf{b}}_i$ are *fixed* while only $\mathbf{t}_i$ changes per signature. This turns the inner products into degree-two relations, enabling preprocessing with only two-party DPFs.

Authenticated VOLE correlation. Given an authenticated secret sharing of y, we want to design a protocol that outputs two vectors of authenticated shares $\mathbf{x} \in \mathbb{Z}_p^{2n}$ and $\mathbf{z} \in \mathbb{Z}_p^{2n}$ such that $\mathbf{z} = \mathbf{x} \cdot y$. Because secret sharing of $(\mathbf{x}, y, \mathbf{z})$ can be viewed as a vector oblivious linear evaluation (VOLE), we call this preprocessing *authenticated VOLE* (aVOLE). Applying aVOLE preprocessing for each value in the trapdoor basis, we can cheaply compute inner products of basis vectors and intermediate projection vectors with online cost linear in the input sizes.

An Efficient PCG for aVOLE. We briefly introduce our improved protocols for generating aVOLE correlations using only two-party DPFs. For general authenticated SPDZ triples, generating the MAC $M[\mathbf{z}] = \Gamma \cdot \mathbf{x} \cdot \mathbf{y}$ requires a degree-three relation: each triplet (i, j, k) must execute a 3-party DPF to compute shares of $\Gamma_k \cdot x_i \cdot y_j$.

For aVOLE, since the input y is already authenticated, $M[y]$ is distributed among the parties, reducing the problem to degree-two. Each pair (i, j) executes a 2-party DPF to compute shares of $x_i \cdot M_j[y]$. Concretely, given $[\![y]\!]$, all ordered pairs run pairwise DPFs to obtain shares of x_i, $x_i \cdot y_j$, $x_i \cdot \Gamma_j$, and $x_i \cdot M_j[y]$. Parties then aggregate the pairwise shares and apply dual-LPN compression to produce the final aVOLE correlation.

Handling precision in the Klein sampler. In each iteration of the Klein sampler, $\mathbf{t}_i$ is an integer vector and the inner product is computed against a fixed-point representation of the orthonormalized basis vector. The inner product loses at most $\log n + 1$ bits of precision (from $2n$ additions) plus 16 bits from the magnitude of $\mathbf{t}_i$ entries. We therefore convert from arithmetic to Boolean shares and discard the $17 + \log n$ least significant bits before computing the corresponding 1D Gaussian. Since the Gaussian output is multiplied by an integral vector from $\mathbf{B}$, the vector $\mathbf{t}_{i-1}$ remains an integer vector and precision resets each iteration.

3 Preliminaries

3.1 Notation

Let λ denote the computational security parameter and log(.) denote logarithms base 2. We use $[a, b] = \{a, \ldots, b\}$ and $[a] = [0, a-1]$. We write $x \leftarrow \mathcal{D}$ to denote sampling x from distribution $\mathcal{D}$, and $x \leftarrow \{0,1\}^n$ for a vector of n uniformly sampled bits. Lowercase bold letters (e.g. $\mathbf{a}$) and upper case letters (e.g. A) denote column vectors, and uppercase bold letters (e.g. $\mathbf{A}$) denote matrices. We use $\mathbb{F}_p$ for the field with p elements, $\mathbb{Z}_p$ for integers modulo p, and $\mathbb{R}$ for reals. We use $\langle \mathbf{a}, \mathbf{b} \rangle$ to denote the inner product of $\mathbf{a}$ and $\mathbf{b}$. We use $\mathcal{B}(p)$ for Bernoulli, $\mathcal{G}_{2^\ell}(p)$ for geometric on $[0, 2^\ell)$, $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$ for the discrete Gaussian on $\mathbb{Z}$, and $D_{\Lambda,\sigma,\mathbf{c}}$ for the discrete Gaussian over lattice Λ centered at $\mathbf{c}$.

For the rest of the paper, $R = \mathbb{Z}[x]/\Phi$ with irreducible polynomial $\Phi = x^n + 1$ denotes a power-of-two cyclotomic ring used in FALCON, where $n = 512$ or 1024 depending on the security level. $R_q = R/qR$ denotes its quotient ring. Note that, for any ring elements $f, g \in R$, there exist the corresponding *anticirculant matrix* representations $\mathcal{A}(f), \mathcal{A}(g) \in \mathbb{Z}^{n \times n}$ such that polynomial addition and multiplication in R can be expressed as matrix addition and multiplication in $\mathbb{Z}^{n \times n}$, i.e., $\mathcal{A}(f+g) = \mathcal{A}(f) + \mathcal{A}(g)$ and $\mathcal{A}(f \cdot g) = \mathcal{A}(f) \cdot \mathcal{A}(g)$. Since we do not exploit ring arithmetic in the signing operations, we slightly abuse the notation and drop $\mathcal{A}(\cdot)$, assuming that ring elements are represented as the anticirculant matrices by default. We present the definition and relevant Lemmas for Rényi divergence from prior works in Appendix A.1.

3.2 Distributions

Definition 1 (Discrete Gaussian Distribution)**.** *The n-dimensional Gaussian function $\rho_{\sigma,\mathbf{c}} : \mathbb{R}^n \to [0, 1)$ centered at $\mathbf{c} \in \mathbb{R}^n$ with standard deviation $\sigma > 0$ is defined by*

$$\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}\right)$$

For any lattice $\Lambda \subseteq \mathbb{R}^n$, the discrete Gaussian distribution $D_{\Lambda,\sigma,\mathbf{c}}$ over Λ centered at $\mathbf{c}$ with variance σ^2 is defined by the probability mass function

$$D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$$

where $\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{y} \in \Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{y})$.

As a special case, when $n = 1$ and $\Lambda = \mathbb{Z}$ (i.e., one-dimensional integer lattice), we write $x \sim \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$ instead of $D_{\mathbb{Z},\sigma,\mu}(x)$ for $\mu \in \mathbb{R}$.

Definition 2 (Geometric Distribution)**.** *If $y \sim \mathcal{G}_{\mathbb{Z}^+}(p)$ with $p \in (0, 1)$, then the probability mass function is*

$$f_y(x) = (1-p)p^x \text{ for } x \in \mathbb{Z}^+.$$

Definition 3 (Bernoulli Distribution)**.** *If $y \sim \mathcal{B}(p)$, then $\Pr[y = 1] = p$ and $\Pr[y = 0] = 1 - p$.*

3.3 Preliminary Algorithms

In this section, we describe preliminary algorithms, including the Klein sampler and MPC-friendly samplers for Geometric and Bernoulli.

Algorithm 1 Sign$(m, sk, \lfloor\beta^2\rfloor)$ [DLP14].

Require: Message m, secret key $sk = \mathbf{B} \in \mathbb{Z}^{2n\times 2n}$, a bound $\lfloor\beta^2\rfloor$
Ensure: Signature `sig` of m

1: **repeat**
2: $r \leftarrow \{0,1\}^{320}$
3: $\mathbf{t} \leftarrow$ HashtoPoint$(r||m, q, n)$
4: $\mathbf{z} \leftarrow$KleinSampler$(\mathbf{B}, \sigma, (\mathbf{t}, 0))$ ▷ $\sigma = 1.55\sqrt{q}$
5: $(\mathbf{s}_1, \mathbf{s}_2) = ((\mathbf{t}, 0) - \mathbf{z})$
6: **until** $||\mathbf{s}_1||^2 + ||\mathbf{s}_2||^2 \leq \lfloor\beta^2\rfloor$
7: return `sig` $= (r, \mathbf{s}_2)$

For key generation, we use `NTRUGen` from the FALCON specification [PFH+20b] (hereafter, the spec) and then use the Gram-Schmidt orthogonalization algorithm (Algorithm 8 in Appendix C) to generate the secret key values required by our signing protocol.

We use a variant of the FALCON signing algorithm [DLP14] as described in Algorithm 1 that uses the Klein sampler for pre-image sampling instead of the `ffSampler` from the spec. Since pre-image sampling is the only part that requires private, non-linear computation, we focus on the Klein sampler further. Note that Klein sampler is not used in FALCON due to high space and time complexity while its security in fact relies on an analysis of Klein presented in [Pre17].

Algorithm 2 describes the Klein sampler, which samples a lattice vector by iteratively projecting the target onto the Gram–Schmidt basis, sampling a 1D discrete Gaussian in each direction, and updating the residual. Other than the secret basis $\mathbf{B} = [[\bar{g}, -\bar{f}], [\bar{G}, -\bar{F}]]$, it uses the Gram-Schmidt orthogonalized basis (GSO) obtained using Algorithm 8. We consider the GSO basis a part of the secret key. Note that the step 6 in Klein sampler calls a 1-D Gaussian sampler with private real center and standard deviation. Thus, in this work we design an efficient MPC-friendly algorithm for 1-D Gaussian sampling with private fixed-point center and standard deviation.

Algorithm 2 KleinSampler$(\mathbf{B}, \sigma, \mathbf{t})$ [Pre16]

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \in \mathbb{Z}^{n\times m}$, its GSO (Alg. 8) $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\}$, a standard deviation σ, a target $\mathbf{t} \in \mathbb{R}^m$
Ensure: $\mathbf{v}$ sampled from $D_{\Lambda(\mathbf{B}),\sigma,\mathbf{t}}$

1: $\mathbf{t}_n \leftarrow \mathbf{t}$
2: $\mathbf{v}_n \leftarrow \mathbf{0}$
3: **for** $i \leftarrow n, \ldots, 1$ **do**
4: $d_i \leftarrow \langle \mathbf{t}_i, \tilde{\mathbf{b}}_i / ||\tilde{\mathbf{b}}_i||^2 \rangle$
5: $\sigma_i \leftarrow \sigma / ||\tilde{\mathbf{b}}_i||$
6: $z_i \leftarrow \mathcal{N}_{\mathbb{Z}}(d_i, \sigma_i^2)$
7: $\mathbf{t}_{i-1} \leftarrow \mathbf{t}_i - z_i \mathbf{b}_i$ ▷ x bits mult.
8: $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i + z_i \mathbf{b}_i$
9: **return** $\mathbf{v_0}$

Like the Klein sampler, `ffSampler` samples a Gaussian with fractional center and standard deviation $\sigma \in [\sigma_{\min}, \sigma_{\max}]$. The FALCON spec presents it as `SamplerZ` (Algorithm 4, Appendix B) sketched in the technical overview. We instead adopt an MPC-friendly sampler built from Bernoulli(e^{-x}) (Wei et al. [WYF+23]) and Geometric(p) (Dwork et al. [DKM+06]) subroutines , shown in Algorithm 6 and Algorithm 7 (Appendix B), because these routines require bit-manipulation, which is friendly to

Functionality $\mathcal{F}_{\mathsf{ThFALCON}}$

Parameters. A monic polynomial $\Phi \in \mathbb{Z}[x]$, a modulus q, fixed-point precision f, $\sigma_{\min}$.
Initialize: On receiving $(\mathsf{init}, \mathrm{sid})$ from all parties:

1. Run $\bar{f}, \bar{g}, \bar{F}, \bar{G} \leftarrow \mathsf{NTRUGen}(\Phi, q)$ and set $\mathbf{B} = \left[\begin{smallmatrix} \bar{g} & -\bar{f} \\ \bar{G} & -\bar{F} \end{smallmatrix}\right]$.
2. Execute $\mathsf{Auth}(\bar{g}, p, n)$, $\mathsf{Auth}(\bar{f}, p, n)$, $\mathsf{Auth}(\bar{G}, p, n)$, $\mathsf{Auth}(\bar{F}, p, n)$ to distribute $[\![\mathbf{B}]\!]$.
3. Compute $\tilde{\mathbf{B}} \leftarrow \mathsf{GSO}(\mathbf{B})$; for $i \in [2n]$, compute $\hat{\mathbf{b}}_i = \frac{\tilde{\mathbf{b}}_i}{\|\tilde{\mathbf{b}}_i\|^2} 2^f$ and execute $\mathsf{Auth}(\hat{\mathbf{b}}_i, p, n)$.
4. For $i \in [2n]$, compute $\mathsf{ccs}_i = \frac{\sigma_{\min}}{\sigma_i} 2^{f-16}$ and execute $\mathsf{Auth}(\mathsf{ccs}_i, 2, f-16)$.
5. Compute and publish $h \leftarrow \bar{g}\bar{f}^{-1} \bmod q$.

Sign: On receiving $(\mathsf{sign}, \mathrm{sid}, \mathrm{ssid}, m)$ from all parties, where Initialize has completed and ssid has not been used in a previous Sign invocation:

1. Sample $r \leftarrow \{0,1\}^{320}$.
2. Obtain $\mathbf{t} \leftarrow \mathtt{HashtoPoint}(r \| m, q, n)$.
3. Use Algorithm 2 with our Gaussian sampling Algorithm 5 to obtain $\mathbf{v} \leftarrow \mathcal{D}_{\Lambda(\mathbf{B}), \sigma, (\mathbf{t} \| 0^n)}$.
4. Compute $(\mathbf{s}_1, \mathbf{s}_2) = ((\mathbf{t}, 0) - \mathbf{v})$.
5. If $||\mathbf{s}_1||^2 + ||\mathbf{s}_2||^2 > \lfloor \beta^2 \rfloor$, go back to step 1.

Output $(\mathsf{sign}, \mathrm{sid}, \mathrm{ssid}, r, \mathbf{s}_2)$ to all parties.

Figure 2: **FALCON signing functionality.** Initialize (called once) generates the trapdoor key material using $\mathsf{NTRUGen}$ and distributes authenticated shares; Sign (called per message with a fresh ssid) produces the signature, allowing concurrent signing sessions. Security guarantees: (1) Unforgeability: an adversary corrupting fewer than t parties cannot produce a valid signature on any unqueried message (see Remark 1); (2) Privacy: partial signatures and key material remain hidden from the adversary, who learns only the final output $(r, \mathbf{s}_2)$.

Boolean MPC.

3.4 MPC Preliminaries

Conventions. N denotes the total number of parties and $\mathcal{P}_i$ denotes party i, $\mathcal{H}$ and $\mathcal{M}$ denote the sets of honest and malicious parties, $\mathcal{S}$ denotes the simulator, and $\mathcal{A}$ the adversary. Ideal functionalities, protocols, and procedures are denoted by $\mathcal{F}_.$, $\Pi_.$, and **procedure**.

Given $x \in \mathbb{Z}_p$, let $[\![x]\!]$ denote authenticated arithmetic shares of x. For $x \in [0, 2^f)$, let $\langle\langle x \rangle\rangle_f$ denote authenticated Boolean shares of x, that is, each of the f bits of x is secret-shared over $\mathbb{Z}_2$. Let $\langle\langle x[i] \rangle\rangle$ denote the Boolean share of the i-th bit of x, so $\langle\langle x \rangle\rangle_f = (\langle\langle x[0] \rangle\rangle, \ldots, \langle\langle x[f-1] \rangle\rangle)$. Given $x \in \mathbb{R}$ represented as a fixed-point value, $\langle\langle x \rangle\rangle_{i.f}$ represents the Boolean shares of the fixed-point representation of x with i and f bits for the integer and fractional part respectively. We use $\langle\langle x \rangle\rangle_{.f}$ as a shorthand for $\langle\langle x \rangle\rangle_{0.f}$.

Security Model. We consider an N-party protocol against a static, active adversary corrupting up to $N-1$ parties, and prove security via the ideal/real paradigm [Can01]. We formalize threshold-signature security via an ideal signing functionality $\mathcal{F}_{\mathsf{ThFALCON}}$ (Figure 2), which captures both unforgeability and privacy of partial signatures: an adversary corrupting fewer than t parties learns nothing beyond the final output signature and cannot produce a valid signature on an unqueried message. This $\mathcal{F}_{\mathsf{ThFALCON}}$-style approach is used in recent threshold signature works including

threshold ECDSA [DKLS25], threshold Schnorr [Lin24], threshold MAYO [BCd+25], and threshold ML-DSA [BdCE+26].

We work in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model (Figure 9, Appendix A.2), which provides standard MPC operations over authenticated secret shares. The functionality supports multiplication, addition, AND, XOR gates, as well as share conversion between arithmetic ($[\![\cdot]\!]$) and Boolean ($\langle\!\langle\cdot\rangle\!\rangle$) representations via A2B and B2A. The functionality also allows sampling from a Bernoulli distribution with public/private probability ρ with κ bit precision, which is simply done by sampling a random value in $[0, 2^\kappa)$ and comparing against the probability $x = 2^\kappa \cdot \rho$. This functionality can be instantiated using SPDZ-style protocols [DPSZ12] in the preprocessing model.

Authenticated Secret Sharing. We use SPDZ-style [DPSZ12] authenticated shares in both arithmetic and Boolean settings. Each party $\mathcal{P}_i$ holds global MAC keys $\Delta_i \leftarrow \{0,1\}^\lambda$ and $\Gamma_i \leftarrow \mathbb{F}_p$, where p is the prime modulus used in the arithmetic setting. For $x \in \{0,1\}$, $\langle\!\langle x \rangle\!\rangle_i := (x_i, M_i[x])$ such that $x = \bigoplus_{i\in[N]} x_i$ and $x \cdot (\bigoplus_{i\in[N]} \Delta_i) = \bigoplus_{i\in[N]} M_i[x]$. Similarly, for $x \in \mathbb{F}_p$, $[\![x]\!]_i := (x_i, M_i[x])$ such that $x = \sum_{i\in[N]} x_i$ and $x \cdot (\sum_{i\in[N]} \Gamma_i) = \sum_{i\in[N]} M_i[x]$.

Distributed Point Function. A two-party Distributed Point Function (DPF) [GI14, BGI16] allows a dealer to split a point function $f_{\alpha,\beta} : \{0,1\}^n \to \mathcal{G}$ (which evaluates to $\beta \in \mathcal{G}$ at input α and to 0 elsewhere) into two compact keys (k_0, k_1). Each key individually reveals nothing about (α, β): for any input x, each party $\mathcal{P}_b$ can locally compute $y_b = \mathsf{Eval}(b, k_b, x)$ such that $y_0 + y_1 = f_{\alpha,\beta}(x)$. Figure 11 and Figure 13 in the Appendix present the ideal functionality and protocol for malicious two-party DPF with one-bit leakage from [ZGY+24]. The DPF functionality allows one bit of leakage via the predicate P. In our construction, the DPF outputs are used as LPN secrets in the PCG. Under the dual-LPN assumption with static leakage, this one-bit leakage does not compromise security.

Dual-LPN and Pseudorandom Correlation Generators. Our protocols operate in the preprocessing model, where an input-independent offline phase generates correlated randomness (e.g., multiplication triples, random authenticated shares) that is used during the online phase. A key tool for efficient preprocessing is the Pseudorandom Correlation Generator (PCG) [BCG+19, BCG+20], which allows the parties to locally expand short correlated seeds into long instances of a target correlation (such as OT correlations or multiplication triples) with sublinear communication.

Definition 4 (Dual-LPN [BCG+19])**.** *Let $\mathcal{C}$ be a probabilistic code generation algorithm such that $\mathcal{C}^\tau(N, k, \mathbb{F})$ outputs a matrix $\mathbf{H} \in \mathbb{F}^{k\times N}$. We say that the decisional dual-LPN(k, N, t) problem is (T, ϵ)- computationally hard for every probabilistic distinguisher $\mathcal{D}$ running in time T if*

$$|\Pr[\mathcal{D}(\mathbf{H}, \mathbf{H}\cdot\mathbf{e}) = 1] - \Pr[\mathcal{D}(\mathbf{H}, \mathbf{u}) = 1]| \le \epsilon|$$

where $\mathbf{e} \leftarrow \mathbb{F}^N$ with weight t and $\mathbf{u} \leftarrow \mathbb{F}^k$.

PCG constructions based on dual-LPN achieve efficient expansion by distributing shares of a sparse noise vector $\mathbf{e}$ via DPF keys. Each party locally computes their share of $\mathbf{He}$, resulting in PCG constructions for OT and multiplication triple correlations with polylogarithmic seed size [BCG+19, BCG+20].

4 MPC-Friendly SamplerZ

Algorithm 4 describes `SamplerZ` from FALCON adapted from [HPRR20], which samples discrete Gaussians. We describe our sampler in Algorithm 3, where instead of directly sampling from $\mathcal{N}_{\mathbb{Z}^+}(0, \sigma^2_{\max})$ in the first step we sample from the geometric distribution $\mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$ and adjust the acceptance probability. Theorem 1 states that both samplers are equivalent; we provide the proof below.

Algorithm 3 Our Algorithm for `SamplerZ`

Require: $\mu \in [0,1]$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$. Let $\tau = 0.5$
Ensure: $z \leftarrow \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$

while true **do**
  $z_0 \leftarrow \mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$
  $b \leftarrow \{0,1\}$
  $z = (2b-1)z_0 + b$
  $x = \frac{(z-\mu)^2}{2\sigma^2} - z_0 \cdot \tau + \frac{\sigma_{\max}^2 \cdot \tau^2}{2}$ ▷ $\frac{(z-\mu)^2}{2\sigma^2} - \frac{z_0^2}{2\sigma_{\max}^2} + \frac{(z_0 - \sigma_{\max}^2 \cdot \tau)^2}{2\sigma_{\max}^2}$
  if $\mathcal{B}(\frac{\sigma_{\min}}{\sigma}) \cdot \mathcal{B}(e^{-x})$ **then**
    Return z

Theorem 1. *Algorithm 3 outputs $z \sim \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$, where $\mu \in [0,1)$, using $\mathcal{G}_{\mathbb{Z}^+}(.)$ and $\mathcal{B}(.)$.*

Proof. We know that Algorithm 4 (from [HPRR20]) outputs $z \sim \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$. Note that Algorithm 3 replaces the base sampler step $y \sim \mathcal{N}_{\mathbb{Z}^+}(0, \sigma_{\max}^2)$ in Algorithm 4 with $z_0 \leftarrow \mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$ if $\mathcal{B}(e^{-\frac{(z_0 - \sigma_{\max}^2 \cdot \tau)^2}{2\sigma_{\max}^2}})$ is true.

Thus, below we show that if $c = \mathcal{B}(e^{-\frac{(z_0 - \sigma_{\max}^2 \cdot \tau)^2}{2\sigma_{\max}^2}}) = 1$ where $z_0 \leftarrow \mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$ then $z_0 \leftarrow \mathcal{N}_{\mathbb{Z}^+}(0, \sigma_{\max}^2)$.

$z_0 \leftarrow \mathcal{G}_{\mathbb{Z}^+}(e^{-\tau})$ implies for any $y \in \mathbb{Z}^+$, $\mathbb{P}[z_0 = y] = (1 - e^{-\tau})e^{-y\tau}$. Thus,

$$\mathbb{E}[c] = \mathbb{E}_{z_0}[e^{-\frac{(z_0 - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2}}]$$

$$= (1 - e^{-\tau}) \sum_{z_0 \in \mathbb{Z}^+} e^{-\frac{(z_0 - \sigma_{\max}^2 \cdot \tau)^2}{2\sigma_{\max}^2}} \cdot e^{-z_0 \tau}$$

$$\mathbb{E}[c] = (1 - e^{-\tau}) e^{-\sigma_{\max}^2 \tau^2/2} \sum_{y \in \mathbb{Z}^+} e^{-\frac{y^2}{2\sigma_{\max}^2}}$$

For any $z_0 \in \mathbb{Z}^+$ conditioned on $c = 1$ (i.e., z_0 being the output),

$$\mathbb{P}[z_0 = y | c = 1] = \frac{\mathbb{P}[c = 1 | z_0 = y]\mathbb{P}[z_0 = y]}{\mathbb{P}[c = 1]}$$

$$= \frac{e^{-\frac{(y - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2}} (1 - e^{-\tau}) e^{-y\tau}}{\mathbb{E}[c]}$$

$$= \frac{e^{-\frac{(y - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2}} (1 - e^{-\tau}) e^{-y\tau}}{(1 - e^{-\tau}) e^{-\sigma_{\max}^2 \tau^2/2} \sum_{y' \in \mathbb{Z}^+} e^{-\frac{y'^2}{2\sigma_{\max}^2}}}$$

$$= \frac{e^{-y^2/2\sigma_{\max}^2}}{\sum_{y' \in \mathbb{Z}^+} e^{-\frac{y'^2}{2\sigma_{\max}^2}}}$$

Thus, z_0 conditioned on $c = 1$ follows the distribution $\mathcal{N}_{\mathbb{Z}^+}(0, \sigma_{\max}^2)$. □

From the above proof it follows that $\mathcal{G}_{M-1}(e^{-\tau})$ leads to $\mathcal{N}_M(\mu, \sigma^2)$.

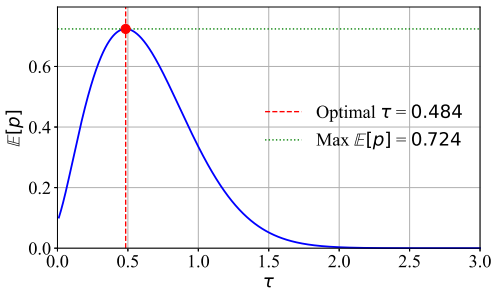


Figure 3: **Acceptance probability for** $\mathcal{N}_{M^+}(0, \sigma_{\max}^2)$ for $M = 2^5$.

4.1 Acceptance Probability

Here, we discuss the choice of τ for optimal circuit size, i.e., τ should maximize the acceptance probability for $\mathcal{N}_{2^\ell}(0, \sigma_{\max}^2)$ from $\mathcal{G}_{2^\ell}(e^{-\tau})$. The acceptance probability is $p = e^{-\frac{(z_0 - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2}}$. The expected value of p is

$$\mathbb{E}[p] = \frac{\sum_{z_0 \in [2^\ell]} e^{-z_0\tau} \cdot e^{-\frac{(z_0 - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2}}}{\sum_{z_0 \in [2^\ell]} e^{-z_0\tau}}$$

Expanding the exponent in the numerator term $-z_0\tau - \frac{(z_0 - \sigma_{\max}^2 \tau)^2}{2\sigma_{\max}^2} = -\frac{z_0^2}{2\sigma_{\max}^2} - \frac{\sigma_{\max}^2 \tau^2}{2}$. Thus,

$$\mathbb{E}[p] = \frac{\sum_{z_0 \in [2^\ell]} e^{-\frac{z_0^2}{2\sigma_{\max}^2}}}{\sum_{z_0 \in [2^\ell]} e^{-z_0\tau}} \cdot e^{-\frac{\sigma_{\max}^2 \tau^2}{2}}$$

Figure 3 plots a curve for the expected value of the acceptance probability vs τ. We maximize $\mathbb{E}[p]$ for $\ell = 5$ at $t = 0.4843$ and in order to avoid an extra multiplication in step 5 of Algorithm 5 we use $\tau = 0.5$ for the acceptance probability $p = 0.72$.

5 Precision Analysis

In this section, we analyze the precision of the Klein sampler and our `SamplerZ` (Algorithm 3) to determine the geometric range and the fixed-point precision needed for Bernoulli-exp (Algorithm 6) and the GSO basis. We first bound the truncation error from the tail cutoff (Lemma 1), then bound fixed-point representation error (Lemma 2), combine the two effects (Lemma 3), and finally lift the bound to the full Klein sampler (Lemma 4). We then use the Rényi divergence bounds from [FGdG+24] (Table 1) to derive the required precision for $\lambda = 128$ and 256.

Lemma 1. ***(Truncated Gaussian).*** *Let $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$ be discrete Gaussian with $\mu \in [0, 1)$ and $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ and $\mathcal{N}_M(\mu, \sigma^2)$ be truncated discrete Gaussian output by Algorithm 3 up to M. Then for q queries,*

$$R_a(\mathcal{N}_M(\mu, \sigma^2) || \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)) \leq 1 + 2q \cdot e^{-M^2/2\sigma_{\max}^2}$$

Lemma 1 can be proven using the Gaussian tail bound with Lemma 12. Thus, we update Algorithm 3 to use $\mathcal{G}_{M-1}(e^{-\tau})$.

Lemma 2. ***(Fixed-point truncated Gaussian vs truncated Gaussian).*** *Let $\mathcal{N}_M(\mu, \sigma^2)$ be truncated discrete Gaussian with reals $\mu \in [0, 1)$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ and $\mathcal{N}'_M(\mu', \sigma'^2)$ be the output of*

Algorithm 3 with fixed-point representation such that $|\mu' - \mu| \leq 2^{-\tilde{f}}$, $|(1/2\sigma^2)' - (1/2\sigma^2)| \leq 2^{-f}$, $|(\frac{\sigma_{\min}}{\sigma}) - (\frac{\sigma_{\min}}{\sigma})'| \leq 2^{-f}$. *Then,*

$$R_a(\mathcal{N}'_M(\mu', \sigma'^2)||\mathcal{N}_M(\mu, \sigma^2))$$

$$\leq 1 + \frac{a(\frac{2^{-\tilde{f}}M}{\sigma^2_{\min}} + (M^2 + M + 1)2^{-f} + \frac{2^{-f}\sigma}{\sigma_{\min}})^2}{2}$$

Proof outline. The ratio $\mathcal{N}'_M(z)/\mathcal{N}_M(z)$ depends on fixed-point errors in the acceptance probability. We define $\delta_x := |x - x'|$ as the error in the exponent $x = \frac{(z-\mu)^2}{2\sigma^2} - z_0\tau + \frac{\sigma^2_{\max}\tau^2}{2}$. Expanding around the exact parameters and using $|z - \mu| \leq M$, $z_0 \leq M$, and $b_{\max} = 1/2\sigma^2_{\min}$, we obtain $\delta_x \leq \frac{2^{-\tilde{f}}M}{\sigma^2_{\min}} + (M^2 + M + 1)2^{-f}$. Combined with the error $\delta_{ccs} \leq 2^{-f}$ in the scaling factor $\sigma_{\min}/\sigma$, we bound $|\mathcal{N}'_M(z)/\mathcal{N}_M(z) - 1| \leq \delta_x + \frac{2^{-f}\sigma}{\sigma_{\min}}$. The Rényi divergence bound then follows from Lemma 13. We defer the complete proof to Appendix E. □

Lemma 3. ***(Fixed-point truncated Gaussian).*** *Let* $\mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$ *be discrete Gaussian with reals* $\mu \in [0, 1)$ *and* $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ *and* $\mathcal{N}'_M(\mu', \sigma'^2)$ *be the truncated output of Algorithm 5 (Appendix B) with fixed-point representation such that* $|\mu' - \mu| \leq 2^{-\tilde{f}}$, $|(1/2\sigma^2)' - (1/2\sigma^2)| \leq 2^{-f}$, $|(\frac{\sigma_{\min}}{\sigma}) - (\frac{\sigma_{\min}}{\sigma})'| \leq 2^{-f}$. *Then for* q *queries,*

$$R_a(\mathcal{N}'_M||\mathcal{N}_{\mathbb{Z}}) \leq (1 + a\frac{\delta^2}{2})(1 + 2qe^{-M^2/2\sigma^2_{\max}})$$

where

$$\delta = \frac{2^{-\tilde{f}}M}{\sigma^2_{\min}} + (M^2 + M + 1)2^{-f} + \frac{2^{-f}\sigma}{\sigma_{\min}}$$

Proof. By multiplicativity (Lemma 4.1 [LSS14]) from Lemma 1 and 2. □

Lemma 4. ***(Fixed-point Klein Sampler).*** *Let* $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ *be the exact KleinSampler and* $\mathcal{D}'_{\Lambda,\sigma,\mathbf{c}}$ *be the fixed-point Klein Sampler for an* n*-dimensional lattice* Λ*, where all* $\hat{\mathbf{b}_i} = \tilde{\mathbf{b}}_i/||\tilde{\mathbf{b}}_i||^2$ *is known up to* f *bits of precision,* $\mathbf{c}$ *is represented by* g *bits and* $1/2\sigma_i^2$ *in known up to* $f - g$ *fractional bits, and* $\mathcal{G}$ *is truncated up to* $M - 1$*. Then for* q_s *sampler queries,*

$$R_a(\mathcal{D}'||\mathcal{D}) \leq 1 + n \cdot (a \cdot \frac{\delta_x^2}{2} + 2n \cdot q_s \cdot e^{\frac{-M^2}{2\sigma^2_{\max}}} + n \cdot q_s \cdot e^{\frac{-M^2}{2\sigma^2_{\max}}} a \cdot \delta_x^2)$$

where,

$$\delta_x = \frac{n2^{-(f-g)}M}{\sigma^2_{\min}} + (M^2 + M + 1)2^{-(f-g)} + \frac{2^{-(f-g)}\sigma}{\sigma_{\min}}$$

Proof. Let us define $Q_i(\cdot|z_{>i})$ and $P_i(\cdot|z_{>i})$ to be conditional distributions for z_i given the previous samples.

We note that in each iteration $\mathbf{v}_i$ is such that $\mathbf{v}_i \leftarrow D_{\Lambda_i,\sigma,c}$ where Λ_i is the $i-th$ projected lattice and $\mathbf{t}_i = \mathbf{c} - \mathbf{v}_i$. Thus, by Lemma 14 we can say that the integer part of d_i can be represented in g bits and by Lemma 15 we can say that each value $\mathbf{t}_i$ can be represented in g bits.

For the $i - th$ step, given $z_{>i}$ the error in d_i can be bounded by $\delta_{d_i} \leq n2^{-f+g}$.

Parameter	$n = 1024, \lambda = 128$	$n = 2048, \lambda = 256$
Rényi Order	72.96	157.05
Maximum Signing Queries (Q_s)	2^{64}	2^{64}
Maximum repetitions C_s	$2^{64} + 2^{50}$	$2^{64} + 2^{36}$
Final Bit security	113	256
Fixed-point precision f	72	73
δ_x (Bits for $\mathcal{B}(e^{-x})$)	2^{-41}	2^{-42}

Table 1: **Parameters for Falcon$^+$-n/2.** Rényi order, signing bounds, and bit security are from [FGdG+24]. Fixed-point precision f and δ_x are derived using Lemma 4 with $M = 32$ ($\ell = 5$ bits for the geometric distribution).

Thus, the Rényi Divergence due to shifted Gaussian using Lemma 3:

$$R_a(\hat{D}_{\mathbb{Z}_i,\hat{d}_i,\sigma_i} || D_{\mathbb{Z}_i,d_i,\sigma_i}) \leq (1 + a\frac{\delta_x^2}{2})(1 + 2n \cdot q_s \cdot e^{\frac{-M^2}{2\sigma_{\max}^2}})$$

where δ_x is as defined earlier. The term $2nq_s$ reflects the total number of 1-D Gaussian calls across all q_s signatures. By Proposition 1(Prop. 4 in [HPRR20]) the total Rényi Divergence satisfies:

$$R_a(P||Q) \leq \prod_{i\in[n]} (1 + a\frac{\delta_x^2}{2})(1 + 2n \cdot q_s \cdot e^{\frac{-M^2}{2\sigma_{\max}^2}})$$

□

For Q_s queries to the signing algorithm, the Klein sampler is called at most C_s times (accounting for repetitions), thus

$$R_a(\mathcal{D}'||\mathcal{D}) \leq 1 + 1/4C_s$$

This is the target bound from [FGdG+24]. With Lemma 4 and Rényi order a from Table 1([FGdG+24]) and given $g = 16$ [PFH+20b] (as the spec outputs 16 bit values for the hash), we get $f = 72$ for Falcon-512, $\lambda = 128$ and $f = 73$ for Falcon-1024, $\lambda = 256$ for $M = 2^5 + 1$.

6 Threshold FALCON

6.1 Functionalities

In this section, we describe the key generation and signing functionalities for threshold FALCON and show how signing reduces to MPC sampling via the Klein sampler. The key-generation functionality commits all secret-key coefficients and auxiliary values (normalized Gram–Schmidt vectors, ccs_i) as authenticated shares and publishes the public key. Signing then hashes the salt and the message, invokes the Klein sampler on the resulting hash, and opens the resulting vector to compute a short signature, so the only secret-key-dependent step is the sampler itself. Because every other computation can be carried out in public, we focus the remainder of the section on the MPC Klein sampler and its primitive samplers, with parameters chosen to satisfy the precision analysis from Section 5.

Key generation. The Initialize step of $\mathcal{F}_{\mathsf{ThFALCON}}$ (Figure 2) runs $\mathsf{NTRUGen}$ to obtain the secret key polynomials $\bar{f}$, $\bar{g}$, $\bar{G}$, $\bar{F}$ satisfying $\bar{f}\bar{G} - \bar{g}\bar{F} \equiv q \bmod \Phi$, distributes authenticated shares of the polynomial coefficients over $\mathbb{F}_p$, computes the Gram-Schmidt basis $\tilde{\mathbf{B}}$ (Algorithm 8), normalizes its entries to fixed-point values shared over $\mathbb{F}_p$, computes $\mathsf{ccs}_i = \sigma_{\min}/\sigma_i$ to $f - 16$ bits and shares

them as Boolean shares over $\mathbb{F}_2^{f-16}$, and publishes the public key $h = \bar{g}\bar{f}^{-1} \bmod q$. We note that a better-than-generic distributed protocol for key generation would require solving the NTRU equation in MPC, which is a distinct open problem left for future work.

Signing. $\mathcal{F}_{\mathsf{ThFALCON}}$ (Figure 2) hashes the salt and message to obtain $\mathbf{t}$, then runs the fixed-precision Klein sampler (Algorithm 2 with Algorithm 5, inner products at $f' = f - \log n - 17$ bits) to produce $\mathbf{v}$, and checks the norm bound. Since all non-sampler steps are public, Figure 5 describes the Klein sampler as an MPC black-box procedure, instantiated with parameters from Section 5.

Concurrent signing. Initialize is called exactly once; key generation must complete before any Sign invocation. After that, multiple Sign sessions can run concurrently, each identified by a fresh sub-session identifier ssid. Sessions are independent: each gets its own fresh salt r and an independent run of the Klein sampler, so concurrent signings share no state beyond the read-only key material $(\mathbf{B}, \tilde{\mathbf{B}})$. We note that our aVOLE preprocessing (Section 7.1) is an optimization for the all-but-one instantiation; to use it, a separate aVOLE preprocessing must be performed for each concurrent signing session, since reusing the same LPN seed across sessions would, after sufficiently many aborts, leak information about the LPN secret (Section 7.1).

6.2 Klein Sampler

The Klein sampler produces a lattice point from a discrete Gaussian by iterating through the Gram–Schmidt basis from $i = 2n - 1$ down to 0. Each iteration computes the inner product $d_i = \langle \mathbf{t}_i, \tilde{\mathbf{b}}_i \rangle / \|\tilde{\mathbf{b}}_i\|^2$, yielding a fixed-point center for the one-dimensional Gaussian. We use arithmetic shares for this computation since these operations are highly *arithmetic*, but as we will see, in the Gaussian sampler it makes more sense to work on *boolean* shares, so we convert $[\![d_i]\!]$ to Boolean shares $\langle\!\langle d_i \rangle\!\rangle$ in order to use the Gaussian procedure. This sampler returns bitwise shares of $z \leftarrow \mathcal{N}_M(d_i, \sigma_i^2)$, which we convert back to arithmetic shares $[\![z]\!]$. The update steps are $\mathbf{t}_{i-1} = \mathbf{t}_i - z \cdot \mathbf{b}_i$ and $\mathbf{v}_{i-1} = \mathbf{v}_i + z \cdot \mathbf{b}_i$ (again, highly arithmetic), accumulating the sampled lattice vector in $\mathbf{v}$. Figure 5 describes the Klein sampler procedure in detail.

Both the inner product and the product $\mathbf{y} = z \cdot \mathbf{b}_i$ contain as a factor a secret-key vector that is input-independent. In Section 7.1 we exploit this structure to generate authenticated triples efficiently when one multiplicand is already fixed and authenticated.

Gaussian sampler. The Gaussian sampler is implemented as a Boolean circuit in $\mathcal{F}_{\mathsf{ABB}}$ and proceeds in three stages: candidate generation, rejection value computation, and acceptance sampling.

Candidate generation. Let the center be $d = v + \mu$, where v is the integer part and μ the fractional part. We obtain the Boolean-shared sample z_0 from the geometric sampler and a random sign bit b, then compute the signed candidate $z = (2b - 1)z_0 + b$. This yields $z = z_0$ when $b = 1$ and $z = -z_0$ when $b = 0$, using an $(\ell + 2)$-bit signed adder.

Rejection value computation. We compute the rejection exponent $x = \frac{(z-\mu)^2}{2\sigma_i^2} - z_0 \cdot \tau + \frac{\sigma_{\max}^2 \cdot \tau^2}{2}$ via a collection of adder and multiplication circuits. The key-dependent term $\frac{1}{2\sigma_i^2}$ is precomputed to $f - 16$ bits during key generation; the values τ and $\frac{\sigma_{\max}^2 \tau^2}{2}$ are public constants. The circuit computes $(z - \mu)^2$ as a fixed-point squaring, multiplies by $\frac{1}{2\sigma_i^2}$, and combines with the remaining terms. Since x requires only 42 bits of precision (from the bound on δ_x in Table 1) and is always positive for $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, we discard excess LSBs and the sign bit.

Acceptance sampling. We sample $y_0 \leftarrow \mathcal{B}(\sigma_{\min}/\sigma_i)$ using precomputed shares and $y_1 \leftarrow \mathcal{B}(e^{-x})$ via the Bernoulli exponent sampler, which takes as input Boolean shares of the exponent. The parties reveal the product $y_0 \cdot y_1$ as the acceptance bit. On acceptance, the output is $z + v$; otherwise, we restart from candidate generation.

Procedure $\mathsf{BerExp}(u)$

On input $(\langle\langle u\rangle\rangle_{a.b}, a, b)$ follow the steps below:

1. Let $p_j = e^{-2^{j-b}}$ for $j \in [a+b]$. Send $(\mathsf{Ber}, 2^\kappa p_j, \kappa)$ to $\mathcal{F}_{\mathsf{ABB}}$ to obtain Boolean shares $\langle\langle x_j\rangle\rangle$ for $j \in [i+b]$.
2. Use $\mathcal{F}_{\mathsf{ABB}}$ to compute $\langle\langle y\rangle\rangle = \wedge_{j\in[a+b]}(\neg\langle\langle u[j]\rangle\rangle \vee \langle\langle x_j\rangle\rangle)$ and output $\langle\langle y\rangle\rangle$.

(a) $\mathcal{B}(e^{-x})$ Bernoulli exponent sampling procedure.

Procedure $\mathsf{Geo}()$

Parameter $q = e^{-0.5}$.

1. Let $p_i = (q^{2^i}/(1+q^{2^i}))$ for each $i \in [\ell]$. All parties send $(\mathsf{Ber}, p_i, \kappa)$ to $\mathcal{F}_{\mathsf{ABB}}$ to obtain bitwise shares $\langle\langle x[i]\rangle\rangle$ such that $x[i] \leftarrow \mathcal{B}(p_i)$ and output $\langle\langle x\rangle\rangle_\ell = \sum_{i\in[\ell]} 2^i\langle\langle x[i]\rangle\rangle$.

(b) $\mathcal{G}(e^{-0.5})$ Geometric sampling procedure.

Procedure $\mathsf{Gauss}(.)$

On input $(\mathsf{Gauss}, \langle\langle d\rangle\rangle_{16.f'}, f', \langle\langle \mathsf{ccs}\rangle\rangle_{.f-16})$ all parties do as follows:

1. Let $\langle\langle v\rangle\rangle_{16}$ and $\langle\langle \mu\rangle\rangle_{.f'}$ be the integer and fractional parts of $\langle\langle d\rangle\rangle_{16.f'}$.
2. Call **procedure** Geo to obtain $\langle\langle z_0\rangle\rangle_\ell$.
3. Sample $\langle\langle b\rangle\rangle \leftarrow \{0,1\}$ and compute $\langle\langle z\rangle\rangle_{\ell+2} = (2\langle\langle b\rangle\rangle - 1)(\langle\langle z_0\rangle\rangle_\ell) + \langle\langle b\rangle\rangle$.
4. Compute $\langle\langle x\rangle\rangle_{13.42} = (\langle\langle z\rangle\rangle_{(\ell+2)} - \langle\langle \mu\rangle\rangle_{.f'})^2\langle\langle \theta_i\rangle\rangle_{.f-16} - \langle\langle z_0\rangle\rangle_\ell(\tau_{.f'}) + (\frac{\sigma_{\max}^2\cdot\tau^2}{2})_{.f'}$.
5. Parties call **procedure** BerExp with $(\langle\langle x\rangle\rangle_{13.42}, 13, 42)$ to obtain $\langle\langle y_0\rangle\rangle$ and $\mathcal{F}_{\mathsf{ABB}}$ with input $(\mathsf{Ber}, \langle\langle \mathsf{ccs}\rangle\rangle_{.f-16}, f-16)$ to obtain $\langle\langle y_1\rangle\rangle$.
6. Parties use $\mathcal{F}_{\mathsf{ABB}}$ to compute $\langle\langle c\rangle\rangle$ and reveal c where $c = y_0y_1$. If $c = 1$ output $\langle\langle d\rangle\rangle_{16}$ where $d = z + v$; otherwise go back to step 2.

(c) Gaussian sampling procedure.

Figure 4: **One-dimensional sampling sub-routines.** a) Instantiates a fixed-point variant of Algorithm 6. b) Instantiates a fixed-point variant of Algorithm 7 for τ. c) The fixed-point MPC procedure for Gaussian sampling from Algorithm 5. We only keep 42 bits for the fractional part of x using precision analysis from Table 1 and 13 bits for the integer part, as the maximum value of the integer part $z - \mu$ is a signed 7-bit number (given $\ell = 5$), thus, we discard the sign bit. We use κ bits for Bernoulli sampling as specified in Table 2.

Geometric sampler. The geometric sampler produces $z_0 \leftarrow \mathcal{G}_{2^\ell}(e^{-\tau})$ by sampling each bit independently. For $i \in [\ell]$, parties compute the public probability $p_i = q^{2^i}/(1+q^{2^i})$ where $q = e^{-\tau}$, then use $\mathcal{F}_{\mathsf{ABB}}$ to sample $x[i] \leftarrow \mathcal{B}(p_i)$. The output is the ℓ-bit integer $z_0 = \sum_{i\in[\ell]} 2^i x[i]$, which is naturally bitwise-shared since we produced each bit independently. Note that since all probabilities are public (determined by $\sigma_{\max}$), geometric samples can be preprocessed before the signing phase.
Bernoulli exponent sampler. To sample from $\mathcal{B}(e^{-x})$ given Boolean shares $\langle\langle x\rangle\rangle_{a.b}$, we decompose the probability as $e^{-x} = \prod_{j\in[a+b]} e^{-2^{j-b}x[j]}$. For each bit position j, parties sample $y_j \leftarrow \mathcal{B}(e^{-2^{j-b}})$

Procedure Klein

Klein Sampler: On input $(\mathsf{Klein}, \mathbf{t}, \left[\!\left[\hat{\mathbf{B}}\right]\!\right], [\![\mathbf{B}]\!], \langle\!\langle \mathsf{ccs} \rangle\!\rangle_{.f-16})$ all parties do as follows: For each $i \in \{2n-1, \ldots, 1, 0\}$ the parties do as follows:

1. If $i = 2n-1$, locally compute $[\![d_i]\!] = \sum_{j\in[2n]} \mathbf{t}[j] \left[\!\left[\hat{\mathbf{b}}_i[j]\right]\!\right]$. Otherwise, use $\mathcal{F}_{\mathsf{ABB}}$ to compute $[\![d_i]\!] = \sum_{j\in[2n]} [\![\mathbf{t}_i[j]]\!] \left[\!\left[\hat{\mathbf{b}}_i[j]\right]\!\right] \in \mathbb{Z}_p$.
2. Use $\mathcal{F}_{\mathsf{ABB}}$ to convert $[\![d_i]\!]$ into boolean shares $\langle\!\langle d_i \rangle\!\rangle_{16.f'}$.
3. Call **procedure** Gauss with input $(\langle\!\langle d_i \rangle\!\rangle_{16.f'}, f', \langle\!\langle \mathsf{ccs}_i \rangle\!\rangle_{.f-16})$ to obtain $\langle\!\langle z \rangle\!\rangle_{16}$, where $z \in \mathbb{F}_{2^{16}}$.
4. Send $(\mathsf{B2A}, \langle\!\langle z \rangle\!\rangle_{16})$ to $\mathcal{F}_{\mathsf{ABB}}$ to obtain $[\![z]\!] \in \mathbb{Z}_p$.
5. Compute $[\![\mathbf{y}]\!] = [\![z]\!] \cdot [\![\mathbf{b}_i]\!]$ using $\mathcal{F}_{\mathsf{ABB}}$.
6. Compute $[\![\mathbf{t}_{i-1}]\!] := [\![\mathbf{t}_i]\!] - [\![\mathbf{y}]\!]$ and $[\![\mathbf{v}_{i-1}]\!] := [\![\mathbf{v}_i]\!] + [\![\mathbf{y}]\!]$, where $\mathbf{t}_{2n-1} = \mathbf{t}$ and $\mathbf{v}_{2n-1} = \{0\}^{2n}$.

All parties output $[\![\mathbf{v}_0]\!]$.

Figure 5: **Klein sampler procedure.** This procedure implements Algorithm 2 with our fixed-point Gaussian sampler (Figure 4).

independently. The output is $y = \bigwedge_{j\in[a+b]}(\neg x[j] \vee y_j)$, which equals 1 with probability e^{-x}. Since the Bernoulli parameters $e^{-2^{j-b}}$ depend only on the bit position, these samples can also be preprocessed.

6.3 Complexity

Each iteration of the Klein sampler involves the following steps. First, the inner product computation requires $O(1)$ rounds and $O(n \log p)$ bits of communication. The subsequent A2B conversion, which computes addition modulo p, requires $O(\log p)$ rounds and $O(\log p)$ bits of communication. The Gaussian sampler computes the exponent x in $O(1)$ rounds with $O(1)$ bits of communication, followed by one round for the acceptance bit and $O(1)$ rounds for the opening; the overall cost scales with the acceptance probability. The B2A conversion opens $16(\log p + 1)$ bits, requiring $O(1)$ rounds and $O(\log p)$ bits of communication. Finally, the product $[\![z]\!]\,[\![\mathbf{b}_i]\!]$ requires $O(1)$ rounds and $O(n \log p)$ bits of communication. Over $2n$ iterations, the total communication complexity is $O(n^2 \log p + n \cdot C)$, where C denotes the cost of the Gaussian sampler per iteration.

6.4 Security

The proofs of the following statements are given in Section E in the Appendix.

Lemma 5 (Geometric Sampler Correctness)**.** π_{Geo} *(Figure 4b) outputs samples from the same distribution as Algorithm 7.*

Lemma 6 (Bernoulli Exponent Correctness)**.** π_{Ber} *(Figure 4a) on input* $\langle\!\langle u \rangle\!\rangle_{a.b}$ *outputs samples from* $\mathcal{B}(e^{-u})$ *matching Algorithm 6.*

Lemma 7 (Gaussian Sampler Correctness)**.** π_{Gauss} *(Figure 4c) outputs samples from* $\mathcal{N}_M(\mu, \sigma^2)$ *matching Algorithm 5.*

Lemma 8 (Klein Sampler Correctness)**.** π_{Klein} *(Figure 5) correctly instantiates Algorithm 2 using* $\mathcal{F}_{\mathsf{ABB}}$*.*

Theorem 2. *Protocol* Π_{sign} *(Figure 6) UC-realizes* $\mathcal{F}_{\mathsf{ThFALCON}}$ *(Figure 2) in the* $\mathcal{F}_{\mathsf{ABB}}$*-hybrid.*

We note that Theorem 2 is secure against any adversarial model that $\mathcal{F}_{\mathsf{ABB}}$ supports. In this work, we focus on security against a static malicious adversary with all-but-one corruption.

Proof structure. The security of our threshold FALCON protocol follows from the following chain of results. Lemmas 5, 6, and 7 establish correctness of the one-dimensional sampling sub-routines; Lemma 8 lifts this to the full Klein sampler; Theorem 2 then establishes that Π_{sign} UC-realizes $\mathcal{F}_{\mathsf{ThFALCON}}$ in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid model, inheriting the security guarantees of the underlying $\mathcal{F}_{\mathsf{ABB}}$. Unforgeability of the threshold scheme then follows from the remark below. The aVOLE optimization (Theorem 3) is a separate result concerning the efficiency of the all-but-one instantiation and is not a dependency of Π_{sign}.

Remark 1 (Unforgeability of Threshold FALCON)**.** $\mathcal{F}_{\mathsf{ThFALCON}}$ *(Figure 2) captures exactly the fixed-point signing algorithm using the Klein sampler with our MPC-friendly* `SamplerZ`*. By Lemma 4, the output distribution of this fixed-point algorithm has bounded Rényi divergence from the ideal discrete Gaussian, so any distinguishing advantage against the fixed-point scheme translates directly into a distinguishing advantage against standard FALCON signing. Since* $\mathcal{F}_{\mathsf{ThFALCON}}$ *only outputs valid FALCON signatures on queried messages and the authenticated key shares of corrupt parties are independent of the secret key, a static active adversary corrupting fewer than* N *parties cannot produce a signature on a fresh message* m^* *without breaking unforgeability of the underlying single-signer FALCON scheme. Thus, security of the threshold protocol reduces to that of FALCON via the Rényi divergence analysis of Section 5. While several game-based unforgeability notions for concurrently secure threshold signatures exist in the literature [BCK+22, CPZ24, BCKT26], to our knowledge, no prior work has positioned the simulation-based security of threshold signature schemes—including ours and [DKLS25, Lin24, BdCE+26]—in the hierarchy of existing threshold unforgeability notions. Intuitively, protocols realizing* $\mathcal{F}_{\mathsf{ThFALCON}}$ *have similar security guarantees to UF-2 [BCK+22, BCKT26], since* $\mathcal{F}_{\mathsf{ThFALCON}}$ *requires all parties to agree on* m *with the same* ssid *to produce a signature. It would be interesting to formally relate the security of* $\mathcal{F}_{\mathsf{ThFALCON}}$ *to existing unforgeability notions for threshold signatures, but we leave this for future work.*

7 All-but-one Malicious Instantiation

Section 6 presents a protocol for threshold FALCON that is agnostic to any MPC-backend. In this section, we describe an optimized instantiation for the all-but-one malicious corruption setting that significantly reduces preprocessing cost.

The key observation is that the Klein sampler's multiplications $z \cdot \mathbf{b}_i$ and inner products $\langle \mathbf{t}_i, \hat{\mathbf{b}}_i \rangle$ always involve the secret key, which is fixed across all signatures. Standard authenticated triple generation requires degree-three relations, but when one multiplicand is already authenticated, we can reduce this to degree-two using *authenticated VOLE*. This makes preprocessing cost sublinear in the number of signatures.

7.1 Authenticated VOLE Protocol

Given N parties with shares of the MAC key Γ, authenticated triple generation requires a degree-three relation to compute the MAC of a product: $M[z] = (\sum_{i\in[N]} \Gamma_i) \cdot (\sum_{j\in[N]} x_j) \cdot (\sum_{k\in[N]} y_k)$, where $z = x \cdot y$. Abram et al. [AS22] propose a 3-party DPF and design a PCG for generating multi-party authenticated triples.

We observe that when one multiplicand is already authenticated—i.e., given shares $(y_i, M_i[y])$ such that $\sum_{i\in[N]} M_i[y] = (\sum_{i\in[N]} y_i) \cdot (\sum_{j\in[N]} \Gamma_j)$—we can rewrite $M[z]$ as a degree-two relation: $M[z] = (\sum_{i\in[N]} M_i[y]) \cdot (\sum_{j\in[N]} x_j)$. We call this *authenticated VOLE*: given $[\![y]\!]$, produce $[\![\mathbf{z}]\!] =$

Protocol Π_{sign}

Parameters. n is the degree of the polynomial. Geometric distribution range parameter 2^{ℓ}, fixed-point error bound 2^{-f}. Constants σ, $\sigma_{\min}$, $\sigma_{\max}$. Arithmetic modulus p prime $> 2^{16+f+1+\log n}$.
Input. Each party $\mathcal{P}_i$ holds MAC keys $\Gamma_i \leftarrow \mathbb{Z}_p$ and $\Delta_i \leftarrow \mathbb{F}_{2^\lambda}$.
Initialize. On input $(\mathsf{init}, \mathrm{sid})$, $\mathcal{P}_i$ uniformly samples $\mathbf{r}_i \leftarrow \{0,1\}^{320}$; All parties use $\mathcal{F}_{\mathsf{ABB}}$ to evaluate $\mathsf{KeyGen}(\oplus_{i\in[N]}\mathbf{r}_i)$ to obtain $[\![\mathbf{B}]\!]$, $\left[\!\left[\hat{\mathbf{B}}\right]\!\right]$, $\langle\!\langle \mathsf{ccs}_j \rangle\!\rangle_{.f-16}$ and $\langle\!\langle \theta_j \rangle\!\rangle_{.f-16}$ for $j \in [2n]$ and the public key h.
Sign. On input $(\mathsf{sign}, \mathrm{sid}, \mathrm{ssid}, m)$:

1. All parties use a crs to sample $r \leftarrow \{0,1\}^{320}$.
2. Obtain $\mathbf{t} \leftarrow \mathsf{HashtoPoint}(r\|m, q, n)$.
3. Send $(\mathsf{Klein}, (\mathbf{t}\|0^n), \left[\!\left[\hat{\mathbf{B}}\right]\!\right], [\![\mathbf{B}]\!], \langle\!\langle \mathsf{ccs} \rangle\!\rangle_{.\kappa})$ to the Klein procedure to obtain $[\![\mathbf{v}]\!]$.
4. All parties execute $(\mathsf{Open}, [\![\mathbf{v}]\!])$ to obtain $\mathbf{v}$.
5. All parties compute $(\mathbf{s}_1, \mathbf{s}_2) = ((\mathbf{t}, 0^n) - \mathbf{v})$.
6. If $||\mathbf{s}_1||^2 + ||\mathbf{s}_2||^2 > \lfloor\beta\rfloor^2$, go back to step 1.

Output $(\mathsf{sign}, \mathrm{sid}, \mathrm{ssid}, r, \mathbf{s}_2)$.

Figure 6: **Protocol for threshold FALCON signing.** The Initialize step executes key generation via $\mathcal{F}_{\mathsf{ABB}}$ and distributes authenticated key shares; Sign supports concurrent sessions via fresh sub-session identifiers ssid.

Functionality $\mathcal{F}_{\mathsf{aVOLE}}$

aVOLE: On receiving $(\mathsf{aVOLE}, [\![y]\!], m)$ from all parties, sample $\mathbf{x} \leftarrow \mathbb{F}_p^m$ and compute $\mathbf{z} = \mathbf{x} \cdot y$. Execute $\mathsf{Auth}(\mathbf{x}, p, m)$ and $\mathsf{Auth}(\mathbf{z}, p, m)$ so all parties obtain $[\![\mathbf{x}]\!]$ and $[\![\mathbf{z}]\!]$.

Figure 7: **Functionality for authenticated VOLE.** This functionality extends $\mathcal{F}_{\mathsf{ABB}}$ from Figure 9, inheriting its MAC key state Γ.

$[\![\mathbf{x}]\!] \cdot [\![y]\!]$ and describe the functionality in Figure 7. In the Klein sampler, we precompute authenticated VOLE correlations for each secret key share $[\![\mathbf{b}_i]\!]$ and $\left[\!\left[\hat{\mathbf{b}}_i\right]\!\right]$, reducing the per-signature preprocessing to sublinear cost.

Protocol overview. At a high level, the protocol proceeds in four stages:

1. *Sparse vector generation (Step 1).* Each party $\mathcal{P}_i$ samples random indices $\boldsymbol{\omega}_i$ and values $\boldsymbol{\beta}_i$, defining a sparse vector $\mathbf{u}_i$ with t non-zero entries.

2. *Pairwise DPF expansion (Step 2a).* Each ordered pair of parties (i, j) runs the maliciously secure 2-party DPF protocol from Zhang et al. [ZGY+24] (Figure 11) to obtain pairwise shares of unit vectors $\mathbf{v}^{(i,j)}$ and correlated vectors $\mathbf{w}^{(i,j)}$ such that $\mathbf{w}^{(i,j)} = r \cdot \mathbf{v}^{(i,j)}$ for uniform $r \leftarrow \mathbb{F}_{2^\lambda}$.

3. *Correction word computation (Steps 2b 2d, 2e).* Parties use $\mathcal{F}_{\mathsf{2PC}}$ to compute correction words $\mathsf{CW}[h]$ that embed the products $\beta_i \cdot \Gamma_j$, $\beta_i \cdot y_j$, and $\beta_i \cdot M_j[y]$. These correction words allow parties to locally compute pairwise shares of $\mathbf{b}^{(i,j)} = \mathbf{u}_i \cdot \Gamma_j$, $\mathbf{c}^{(i,j)} = \mathbf{u}_i \cdot y_j$, and $\mathbf{d}^{(i,j)} = \mathbf{u}_i \cdot M_j[y]$.

4. *Aggregation and LPN compression (Step 3, 4).* Each party $\mathcal{P}_i$ aggregates pairwise shares to obtain $\mathbf{b}_i$, $\mathbf{c}_i$, $\mathbf{d}_i$ such that $\mathbf{b} = \mathbf{u} \cdot \Gamma$, $\mathbf{c} = \mathbf{u} \cdot y$, and $\mathbf{d} = \mathbf{u} \cdot M[y]$. Finally, all parties apply the dual-LPN matrix $\mathbf{H}$ to compress these sparse vectors into random-looking outputs: $\mathbf{x} = \mathbf{H}\mathbf{u}$, $\mathbf{z} = \mathbf{H}\mathbf{c}$, $M[\mathbf{x}] = \mathbf{H}\mathbf{b}$, and $M[\mathbf{z}] = \mathbf{H}\mathbf{d}$.

Figure 8 describes the authenticated VOLE protocol in detail.

Complexity. Each pairwise DPF execution requires $O(\log(\mathbb{N}/t))$ rounds: an initial OT to authenticate the index ω, followed by $\log(\mathbb{N}/t)$ rounds that each open λ bits for correction words, sample random challenges via coin toss, and obtain authenticated random bits via OT. A final MAC check completes the protocol.

In the aVOLE protocol, all parties execute t pairwise DPFs in parallel, incurring $O(N \cdot t \log(\mathbb{N}/t))$ bits of communication per party. The subsequent steps—six unauthenticated multiplications and opening correction words—add $O(N \cdot t \log p)$ bits and $O(1)$ rounds. The protocol concludes with a batched MAC check using a coin toss protocol, requiring $O(1)$ rounds and $O(\log p)$ communication. Since all t DPFs across all party pairs run in parallel, the total round complexity is $O(\log(\mathbb{N}/t))$ and the communication complexity per party is $O(N \cdot t \cdot (\log(\mathbb{N}/t) + \log p))$.

Per signature, the Klein sampler consumes $O(n)$ aVOLE correlations rather than generating $O(n)$ fresh authenticated triples.

The proofs to the following statements appear in Section E.

Lemma 9 (Unforgeability). *If $\mathbf{z} \neq \mathbf{x} \cdot y$, then the batch-check in step 5 of Π_{aVOLE} (verifying $M[\mathbf{x}] = \Gamma \cdot \mathbf{x}$ or $M[\mathbf{z}] = \Gamma \cdot \mathbf{z}$) fails with all but negligible probability.*

Lemma 10 (Correctness). *If all parties are honest, then Π_{aVOLE} (Figure 8) generates the correct correlation specified by command* (aVOLE) *in $\mathcal{F}_{\mathsf{aVOLE}}$ (Figure 7).*

Theorem 3 (Security of Authenticated VOLE). *Assuming decisional dual-LPN is hard with static leakage, protocol Π_{aVOLE} (Figure 8) satisfies simulation-based security realizing $\mathcal{F}_{\mathsf{aVOLE}}$ (Figure 7) in the $(\mathcal{F}_{\mathsf{DPF}}, \mathcal{F}_{\mathsf{2PC}}, \mathcal{H})$-hybrid model against a static malicious adversary, where $\mathcal{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_p$ is a random oracle.*

Remark 2 (UC composition and concurrent sessions). *The simulator in the proof E.3 of Theorem 3 is straight-line and the random oracle $\mathcal{H}$ is never programmed, it is only used as a correlation-robust function. Theorem 3 therefore holds in the $(\mathcal{F}_{\mathsf{DPF}}, \mathcal{F}_{\mathsf{2PC}}, \mathsf{gRO})$-hybrid model, where* gRO *denotes the strict (non-programmable, observable) global random oracle of [CJS14], for which UC composition theorem applies. Thus, Π_{aVOLE} can be composed with itself and with other UC-secure protocols.*

We note that $\mathcal{F}_{\mathsf{DPF}}$ allows the adversary to learn b predicate bits on the DPF index with 2^{-b} probability and after the dual LPN transformation it results in static leakage on the LPN secret for the session. Thus, concurrent signing sessions of protocol 6 should use fresh aVOLE correlations.

8 Evaluation

8.1 Implementation Details

We use the FALCON reference implementation for key generation and verification. Instead of evaluating KeyGen in MPC as in Figure 6, a trusted party distributes authenticated secret key shares and publishes the public key. During setup, the parties run a coin-tossing protocol to obtain the seed for the random hash function used during signing. We implement [2] the online phase of the signing protocol and read the preprocessing material—authenticated VOLEs, Boolean triples, daBits, and Bernoulli samples—from files stored on disk.

We benchmark two instantiations of the signing protocol.

- **3PC honest majority:** a semi-honest three-party protocol using replicated secret sharing for arithmetic and garbled circuits for Boolean computation (round-optimized).

[2] Our implementation is available at https://github.com/radhika1601/threshold-falcon

Protocol Π_{aVOLE}

Parameters: Dual LPN-$(\mathbb{N}, m, t)$, with matrix $\mathbf{H}^T \leftarrow \mathcal{C}(\mathbb{N}, m, \mathcal{R})$. Random oracle $\mathcal{H} : \{0,1\}^{\lambda+2} \to \mathbb{Z}_p$.
Input: Each party $\mathcal{P}_i$ has its MAC key $\Gamma_i \leftarrow \mathbb{Z}_p$, the secret-shared value $[\![y]\!] = (y_i, M_i[y])$, such that $y \in \mathbb{Z}_p$ and $M[y] = \sum_{i\in[N]} M_i[y] = \Gamma \cdot y \bmod p$.
Output: $[\![\mathbf{x}]\!]\,[\![y]\!] = [\![\mathbf{z}]\!]$, $\mathbf{x}, \mathbf{z} \in \mathbb{Z}_p^m$
Protocol:

1. For all $i \in [N]$, $\mathcal{P}_i$ samples $\boldsymbol{\beta}_i \leftarrow \mathbb{Z}_p^t$ and $\boldsymbol{\omega}_i \leftarrow [\mathbb{N}/t]^t$. Then it computes the vector $\mathbf{u}_i$ where for all $k \in [t]$, $(\mathbf{u}_i[\boldsymbol{\omega}_i[k] + \frac{k\mathbb{N}}{t}]) = \boldsymbol{\beta}_i[k]$ and $\mathbf{u}_i[k] = 0$ otherwise.
2. For every $i, j \in [N]$ the pair of parties $\mathcal{P}_i$ and $\mathcal{P}_j$ do as follows for each $k \in [t]$:

(a) $\mathcal{P}_i$ sends $(\mathsf{gen}, \boldsymbol{\omega}_i[k])$ and $\mathcal{P}_j$ sends (gen) to $\mathcal{F}_{\mathsf{DPF}}$ to obtain $(\mathbf{v}_i^{(i,j)}, \mathbf{w}_i^{(i,j)})$ and $(\mathbf{v}_j^{(i,j)}, \mathbf{w}_j^{(i,j)})$ respectively such that $(\mathbf{v}_i^{(i,j)} \oplus \mathbf{v}_j^{(i,j)})[\boldsymbol{\omega}_i[k]] = 1$ and zero otherwise, similarly $(\mathbf{w}_i^{(i,j)} \oplus \mathbf{w}_j^{(i,j)})[\boldsymbol{\omega}_i[k]] = r \in \mathbb{F}_{2^\lambda}$ and zero otherwise.

(b) $\mathcal{P}_i$ and $\mathcal{P}_j$ use (uMult) from $\mathcal{F}_{\mathsf{2PC}}$ to obtain shares of $\alpha[0] = \boldsymbol{\beta}_i[k] \cdot \Gamma_j$, $\alpha[1] = \boldsymbol{\beta}_i[k] \cdot y_j$ and $\alpha[2] = \boldsymbol{\beta}_i[k] \cdot M_j[y]$.

(c) For each $b \in \{i, j\}$, let $s_b := (b == i)$. Each party $\mathcal{P}_b$ computes:

$$\mathsf{A}_b := (-1)^{s_b} \cdot \sum_{l\in[0,\mathbb{N}/t)} \mathbf{v}_b^{(i,j)}[l]$$

$$\mathsf{B}_b[h] := (-1)^{1-s_b} \cdot \sum_{l\in[0,\mathbb{N}/t)} \mathcal{H}(\mathbf{w}_b^{(i,j)}[l]|h)_{h\in[3]}$$

such that $\mathsf{A}_i + \mathsf{A}_j = \mathbf{v}_j^{(i,j)}[\boldsymbol{\omega}_i[k]] - \mathbf{v}_i^{(i,j)}[\boldsymbol{\omega}_i[k]]$, $\mathsf{B}_i[0] + \mathsf{B}_j[0] = \mathcal{H}(\mathbf{w}_i^{(i,j)}[\boldsymbol{\omega}_i[k]]|00) - \mathcal{H}(\mathbf{w}_j^{(i,j)}[\boldsymbol{\omega}_i[k]]|00)$, etc.

(d) Both parties use (uMult) in $\mathcal{F}_{\mathsf{2PC}}$ to get $\mathsf{CW}^{(i,j)}[l] = (\mathsf{A}_i + \mathsf{A}_j) \cdot (\mathsf{B}_i[l] + \alpha_i[l] + \mathsf{B}_j[l] + \alpha_j[l])$ for $l \in [3]$.

(e) $\mathcal{P}_i$ sends $(\mathsf{CW}_i^{(i,j)}[l])_{l\in[3]}$ to $\mathcal{P}_j$ and vice versa. Both parties compute $(\mathsf{CW}^{(i,j)}[l] := \mathsf{CW}_i^{(i,j)}[l] + \mathsf{CW}_j^{(i,j)}[l])_{l\in[3]}$. For $b \in \{i, j\}$, let $s_b := (b == i)$. Each party $\mathcal{P}_b$ computes their share of $\mathbf{b}^{(i,j)}$, $\mathbf{c}^{(i,j)}$, $\mathbf{d}^{(i,j)}$ as follows:

$$\mathbf{b}_b^{(i,j)}[\frac{k\mathbb{N}}{t} + l] = (-1)^{s_b} \cdot (\mathcal{H}(\mathbf{w}_b^{(i,j)}[l]|00) + \mathbf{v}_b^{(i,j)}[l] \cdot \mathsf{CW}^{(i,j)}[0])$$

$$\mathbf{c}_b^{(i,j)}[\frac{k\mathbb{N}}{t} + l] = (-1)^{s_b} \cdot (\mathcal{H}(\mathbf{w}_b^{(i,j)}[l]|01) + \mathbf{v}_b^{(i,j)}[l] \cdot \mathsf{CW}^{(i,j)}[1])$$

$$\mathbf{d}_b^{(i,j)}[\frac{k\mathbb{N}}{t} + l] = (-1)^{s_b} \cdot (\mathcal{H}(\mathbf{w}_b^{(i,j)}[l]|10) + \mathbf{v}_b^{(i,j)}[l] \cdot \mathsf{CW}^{(i,j)}[2])$$

for each $l \in [0, \mathbb{N}/t)$.

3. Each $\mathcal{P}_i$ computes $\mathbf{b}_i = \sum_{i\neq j}(\mathbf{b}_i^{(i,j)} + \mathbf{b}_i^{(j,i)}) + \mathbf{u}_i \cdot \Gamma_i$, $\mathbf{c}_i = \sum_{i\neq j}(\mathbf{c}_i^{(i,j)} + \mathbf{c}_i^{(j,i)}) + \mathbf{u}_i \cdot y_i$ and $\mathbf{d}_i = \sum_{i\neq j}(\mathbf{d}_i^{(i,j)} + \mathbf{d}_i^{(j,i)}) + \mathbf{u}_i \cdot M_i[y]$.
4. Each party $\mathcal{P}_i$ outputs:

$$\mathbf{x}_i = \mathbf{H}\mathbf{u}_i, \quad M_i[\mathbf{x}] = \mathbf{H}\mathbf{b}_i,$$
$$\mathbf{z}_i = \mathbf{H}\mathbf{c}_i, \quad M_i[\mathbf{z}] = \mathbf{H}\mathbf{d}_i.$$

5. All parties execute batched mac-check for $[\![\mathbf{x}]\!]_i = (\mathbf{x}_i, M_i[\mathbf{x}])$ and $[\![\mathbf{z}]\!]_i = (\mathbf{z}_i, M_i[\mathbf{z}])$.

Figure 8: **Protocol for authenticated VOLE generation.**

Parameters	p	f	f'	κ
Value	$2^{100} - 15$	73	45	74

Table 2: **Parameters used for evaluation.** We use the same parameters for both $\log n = 9$ and $\log n = 10$. Thus, $f = 73$. κ here is the number of bits used in the Bernoulli sampling, and $R(\mathcal{B}(x)||\mathcal{B}(x')) \leq 1 + 2^{-66}$ given $|x - x'| \leq 2^{-\kappa}$.

- **All-but-one corruption:** a malicious N-party protocol using SIMD GMW [GMW87] with SPDZ-style authenticated Boolean triples; garbled circuits would be preferable in high-latency settings.

Both settings run on AWS `c7i.large` instances; for the all-but-one setting we benchmark $N \in \{2, 4, 8, 16\}$ parties with network bandwidth up to 10 Gbps and 0.1 ms latency. Table 2 lists parameters for both $\log n = 10$ and 9.

8.2 Summary

We summarize the findings from our evaluation as follows:

- **Online signing:**
  - **3PC honest majority:** For semi-honest setting with honest majority, Falcon-512 signing takes 3.45 s and 16.02 MB of communication amortized over 10 messages and total rounds 4643.
  - **All-but-one corruption:** For $N = 4$ parties, Falcon-512 signing takes 7.66 s and 162.51 MB of communication; Falcon-1024 takes 15.51 s and 572.58 MB. Time and communication per party both scale linearly with the number of parties (Section 8.3).
- **Preprocessing:** Compared to naive MPC execution of the double-precision FFO sampler, our approach reduces the required AND gates by $\sim 13\times$. For $N = 4$ parties, our protocol requires 310.3 s while the reference implementation requires 2618.12 s (Section 8.3).
- **aVOLE efficiency:** Our aVOLE protocol is $13\times$ faster and requires $9{,}800\times$ less communication than standard authenticated triple generation using MASCOT for $N = 4$ parties, due to the sublinear communication complexity of the underlying DPF-based construction (Section 8.4).

8.3 Signing Performance

We benchmark the performance of the signing protocol in two phases: 1) online cost (message dependent) and 2) preprocessing cost (message independent). The preprocessing phase generates Boolean triples, daBits, the Bernoulli preprocessing material, and authenticated VOLE correlations derived from the secret-key shares.

Online signing cost. Table 3 reports the online signing cost for Falcon-512 ($\log n = 9$) and Falcon-1024 ($\log n = 10$) across both security settings. The 3PC honest-majority results are amortized over 10 signatures, whereas the all-but-one results are amortized over 2000 signatures.

In the 3PC honest-majority setting, communication grows as $O(n^2)$, while running time grows as $O(n)$ and is dominated by network latency. The FFO sampler would reduce communication, but not running time, because its $O(n)$ round complexity remains the bottleneck.

In the all-but-one setting, Falcon-1024 takes roughly twice as long as Falcon-512, reflecting the $2n$ iterations of the Klein sampler. Each iteration involves an inner product and a scalar-vector multiplication over $\mathbb{Z}_p^{2n}$, so per-signature communication grows as $O(n^2 \log p)$. Communication

	$\log n = 9$		$\log n = 10$	
N	Time (s)	Comm (MB)	Time (s)	Comm (MB)
3PC honest majority (amortized over 10 *signatures)*				
3	3.46	2131.51	6.97	4281.58
All-but-one corruption (amortized over 2000 *signatures)*				
2	3.65	54.36	9.27	151.23
4	7.66	162.51	15.51	572.58
8	15.46	353.36	27.74	1242.50
16	30.19	800.29	70.04	2808.05

Table 3: **Online signing performance.** Time and communication per signature for both security settings.

	$\log n = 9$			$\log n = 10$		
Protocol	#Corr	Time (s)	Comm (GB)	#Corr	Time (s)	Comm (GB)
Bool. triples	140M	204.4	0.26	236M	344.56	0.42
aVOLE	2.1M	95.5	0.01	8.4M	602.8	0.05
daBits	120K	10.4	2.2	238K	20.7	19.4
Total	—	**310.3**	**2.47**	—	**968.06**	**19.87**

Table 4: **Preprocessing cost per signature.** Number of correlations (#Corr), time, and communication (Comm) per signature for $N = 4$ parties, averaged over 2000 messages. Costs estimated using authenticated triple generation from [YWZ20], daBit generation from MP-SPDZ [Kel20], and aVOLE from Section 8.4.

also scales approximately linearly with the number of parties because the GMW-style protocol is broadcast based. Since signing is probabilistic, both running time and communication vary slightly across executions.

Number of Rounds. The 3PC honest majority implementation is round optimized because it uses garbled circuits for Boolean computation. We observe that the number of rounds is $\approx 4.5n$. The total number of rounds for signing 10 signatures is 4643 and 9281 for $\log n = 9$ and $\log n = 10$, respectively. In contrast, the all-but-one protocol implementation is not round optimized because it uses GMW for the Boolean computation. For the all-but-one setting, we benchmark the total number of rounds for signing 2000 signatures: $\log n = 9$ requires 12,378,038 and $\log n = 10$ requires 20,359,173 rounds. Amortized over 2000 signatures, this is 6,189 rounds per signature for $\log n = 9$ and 10,179 for $\log n = 10$; high-latency environments favor constant-round backends such as garbled circuits.

Preprocessing cost. Table 4 shows the number of correlations required per signature for both $\log n = 9$ and $\log n = 10$. For $N = 4$, we estimate preprocessing costs using: authenticated-triple-generation costs from Tables 2 and 4 of [YWZ20], daBit-generation benchmarks from the MASCOT implementation in MP-SPDZ [KOS16, Kel20], and our aVOLE benchmarks from Section 8.4.

Baseline performance. In this section, we compare against two baselines in the respective circuit size:

- `ffSampler`**:** Naive MPC execution of the original FALCON signing algorithm.
- **Klein:** We replace `ffSampling` with a double-precision Klein Sampler while keeping the Gaussian sampler `SamplerZ` from the specification.

For both baselines, we benchmark the number of private algebraic operations required in Table 5.

Method	$\log n$	Double Add	Double Mult.	Other ANDs	Total ANDs
`ffSampler`	9	76,479	71,857	8.647M	1.83B
`ffSampler`	10	162,635	149,806	17.12M	3.834B
Klein	9	2.124M	1.084M	41.06M	32.765B
Klein	10	8.443M	4.265M	146.9M	129.3B

Table 5: **Estimated cost for the baselines.** We report the number of operations required using the Falcon reference implementation, averaged over 1000 messages. We also report the cost of replacing the specification's `ffSampler` with a double-precision Klein sampler.

	Time (s)		Comm (MB)	
Parties	Ours	Triples	Ours	Triples
2	42.4502	297.136	2.03834	17870.4
4	75.3527	1015.6	6.11502	60422.8
8	142.735	5112.37	14.2684	140964
16	190.095	—	30.5751	—

Table 6: **aVOLE vs. authenticated triple generation.** Cost of generating 2^{20} correlations using our aVOLE protocol versus MASCOT [KOS16] triple generation from MP-SPDZ [Kel20]. aVOLE uses dual-LPN parameters $m = 2^{20}$, $C = 1/7$, $t = 224$ from [SLY+25]. For $N = 4$, aVOLE is 13× faster and requires 9,800× less communication.

Appendix D describes how we do the instrumentation and obtain the reported cost. The Klein sampler's higher cost stems from its $O(n)$ round complexity vs. $O(\log n)$ for `ffSampler`; a fixed-precision analysis for `ffSampler` is an open problem. An alternative is to thresholdize the *hybrid* Klein sampler [Pre15, EFG+22], which is parallelizable and has an offline-online structure that may yield more efficient MPC implementations [ENS+23].

8.4 aVOLE Benchmarks

We implement the aVOLE protocol (Section 7.1) using the malicious 2-party DPF protocol from Zhang et al. [ZGY+24]. We use conservative EA-LPN parameters from prior work [SLY+25]: $m = 2^{20}$ output correlations, code rate $C = 1/7$, row Hamming weight $\rho = 224$, and error weight $t = 224$.

Table 6 compares the cost of generating 2^{20} aVOLE correlations against authenticated triple generation using MASCOT [KOS16] from MP-SPDZ [Kel20]. All experiments run on `r6i.xlarge` instances with 8 threads. For $N = 4$ parties, aVOLE is 13× faster and requires 9,800× less communication than MASCOT. The communication improvement stems from aVOLE's sublinear complexity: each party communicates $O(t \cdot \log(m/t))$ bits per correlation, compared to $O(m)$ for standard triple generation.

Disclaimer

This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or

service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

Acknowledgements

The work of Xiao Wang and Radhika is supported by JP Morgan Research Award and NSF awards #2236819 and #2310927.

References

[ADP24] Nabil Alkeilani Alkadri, Nico Döttling, and Sihang Pu. Practical lattice-based distributed signatures for a small number of signers. In Christina Pöpper and Lejla Batina, editors, *ACNS 2024, Part I*, volume 14583 of *LNCS*, pages 376–402. Springer, Cham, March 2024.

[ALLW25] Martin R. Albrecht, Russell W. F. Lai, Oleksandra Lapiha, and Ivy K. Y. Woo. Partial lattice trapdoors: How to split lattice trapdoors, literally. Cryptology ePrint Archive, Report 2025/367, 2025.

[AS22] Damiano Abram and Peter Scholl. Low-communication multiparty triple generation for SPDZ from ring-LPN. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 221–251. Springer, Cham, March 2022.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.

[BCd+25] Giacomo Borin, Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. Cryptology ePrint Archive, Report 2025/1166, 2025.

[BCdP+26] Giacomo Borin, Sofía Celi, Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. Hermine: An efficient lattice-based FROST-like threshold signature. Cryptology ePrint Archive, Paper 2026/419, 2026.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Cham, August 2019.

[BCG+20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 387–416. Springer, Cham, August 2020.

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022.

[BCKT26] Jan Bobolz, Elizabeth Crites, Markulf Kohlweiss, and Akira Takahashi. Uc4free! existing threshold signatures are uc secure. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026*, pages 63–93, Cham, 2026. Springer Nature Switzerland.

[BdCE+25] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. Efficient, scalable threshold ML-DSA signatures: An MPC approach. Cryptology ePrint Archive, Report 2025/1163, 2025.

[BdCE+26] 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*, 2026.

[BFLP26] Daniel De Almeida Braga, Pierre-Alain Fouque, Bachir Lachguel, and Thomas Prest. Toward a secure fixed-point implementation of the falcon signature scheme. Springer-Verlag, 2026.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[BKL+25] 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 Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 149–164. IEEE Computer Society Press, May 2025.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In Michael J. Jacobson, Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 2013*, volume 7954 of *LNCS*, pages 218–236. Springer, Berlin, Heidelberg, June 2013.

[BLR+18] Shi Bai, Tancrède Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, April 2018.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CATZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 268–302. Springer, Singapore, December 2024.

[CdPE+26] Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Efficient threshold ML-DSA. In *USENIX Security Symposium (USENIX Security 2026)*, 2026.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical uc security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 597–608, 2014.

[CKS20] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

[CPZ24] Cas Cremers, Aleksi Peltonen, and Mang Zhao. An extended hierarchy of security notions for threshold signature schemes and automated analysis of protocols that use them. Cryptology ePrint Archive, Paper 2024/1920, 2024.

[dENP25] Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Simple and efficient lattice threshold signatures with identifiable aborts. Cryptology ePrint Archive, Report 2025/871, 2025.

[DKLS25] Antonín Dufka, Semjon Kravtšenko, Peeter Laud, and Nikita Snetkov. Trilithium: Efficient and universally composable distributed ML-DSA signing. Cryptology ePrint Archive, Report 2025/675, 2025.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 486–503. Springer, 2006.

[DKM+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 22–41. Springer, Berlin, Heidelberg, December 2014.

[DOTT22] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, April 2022.

[DP15] Léo Ducas and Thomas Prest. A hybrid Gaussian sampler for lattices over rings. Cryptology ePrint Archive, Report 2015/660, 2015.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Berlin, Heidelberg, August 2012.

[EEN+24] Muhammed F. Esgin, Thomas Espitau, Guilhem Niot, Thomas Prest, Amin Sakzad, and Ron Steinfeld. Plover: Masking-friendly hash-and-sign lattice signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 316–345. Springer, Cham, May 2024.

[EFG+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 222–253. Springer, Cham, May / June 2022.

[EKT24] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 387–424. Springer, Cham, August 2024.

[ENP24] Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458. Springer, Cham, August 2024.

[ENS+23] Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation - making mitaka as secure as falcon. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 3–36. Springer, Singapore, December 2023.

[FGdG+24] Pierre-Alain Fouque, Phillip Gajland, Hubert de Groote, Jonas Janneck, and Eike Kiltz. A closer look at falcon. Cryptology ePrint Archive, Paper 2024/1769, 2024.

[FW24] Yucheng Fu and Tianhao Wang. Benchmarking secure sampling protocols for differential privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 318–332. ACM Press, October 2024.

[GHK+25] 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.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, Berlin, Heidelberg, May 2014.

[GKS24] Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GTJ+26] Liming Gao, Guofeng Tang, Dingding Jia, Yijian Liu, Bingqian Liu, Xianhui Lu, Kunpeng Wang, and Yongjian Yin. sftalong: Bandwidth-efficient two-round threshold signatures from lattices. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part I*, Lecture Notes in Computer Science, pages 94–123. Springer, 2026.

[HHP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In Marc Joye, editor,

CT-RSA 2003, volume 2612 of *LNCS*, pages 122–140. Springer, Berlin, Heidelberg, April 2003.

[HPRR20] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 53–71. Springer, Cham, 2020.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International algorithmic number theory symposium*, pages 267–288. Springer, 1998.

[JTWZ26] Kaijie Jiang, Stefano Tessaro, Hoeteck Wee, and Chenzhi Zhu. Tweed: Adaptively secure lattice-based two-round threshold signatures. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part I*, Lecture Notes in Computer Science, pages 124–153. Springer, 2026.

[JWZ26] Kaijie Jiang, Hoeteck Wee, and Chenzhi Zhu. Oriole: Adaptively secure partially non-interactive threshold signatures from lattices. Cryptology ePrint Archive, Paper 2026/793, 2026.

[Kel20] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. Cryptology ePrint Archive, Report 2020/521, 2020.

[Kle00] Philip Klein. Finding the closest lattice vector when it's unusually close. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 937–941, 2000.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016.

[KRT24] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 459–491. Springer, Cham, August 2024.

[Lin24] Yehuda Lindell. Simple three-round multiparty Schnorr signing with full simulatability. *CiC*, 1(1):25, 2024.

[LNK+25] Seunghwan Lee, Jaesang Noh, Taejeong Kim, Dohyuk Kim, and Dong-Joon Shin. Actively secure MPC in the dishonest majority setting: Achieving constant complexity in online communication, computation per gate, rounds, and private input size. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part IV*, volume 16003 of *LNCS*, pages 105–139. Springer, Cham, August 2025.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, Berlin, Heidelberg, May 2014.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, October 2004.

[PFH+20a] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[PFH+20b] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, Technical report, National Institute of Standards and Technology, 2020.

[PKN+25] 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 Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 423–456. Springer, Cham, August 2025.

[PN25] Rafaël Del Pino and Guilhem Niot. Finally! A compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 169–199. Springer, Cham, May 2025.

[PPS26] Jai Hyun Park, Alain Passelègue, and Damien Stehlé. THED: Threshold dilithium from FHE. Cryptology ePrint Archive, Paper 2026/638, 2026.

[Pre15] Thomas Prest. *Gaussian sampling in lattice-based cryptography*. PhD thesis, École Normale Supérieure, Paris, 2015.

[Pre16] Thomas Prest. Gaussian sampling in lattice-based cryptography, 2016. https://tprest.github.io/pdf/pub/thesis-thomas-prest.pdf.

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 347–374. Springer, Cham, December 2017.

[Pub24] Federal Information Processing Standards Publication. FIPS 204: Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication https://doi.org/10.6028/NIST.FIPS.204, 2024.

[RRU26] Matthieu Rambaud, Sascha Roth, and Antoine Urban. Threshold signatures as-a-service: Achieving threshold ML-DSA in one online round. Cryptology ePrint Archive, Paper 2026/814, 2026.

[RW19] Dragos Rotaru and Tim Wood. MArBled circuits: Mixing arithmetic and Boolean circuits with active security. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors,

INDOCRYPT 2019, volume 11898 of *LNCS*, pages 227–249. Springer, Cham, December 2019.

[SLY+25] Yunqing Sun, Hanlin Liu, Kang Yang, Yu Yu, Xiao Wang, and Chenkai Weng. Committed vector oblivious linear evaluation and its applications. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25. Association for Computing Machinery, 2025.

[TPCZ24] Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. Efficient lattice-based threshold signatures with functional interchangeability. Cryptology ePrint Archive, Report 2024/1067, 2024.

[WYF+23] Chengkun Wei, Ruijing Yu, Yuan Fan, Wenzhi Chen, and Tianhao Wang. Securely sampling discrete gaussian noise for multi-party differential privacy. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 2262–2276. ACM Press, November 2023.

[YWZ20] Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1627–1646. ACM Press, November 2020.

[ZGY+24] Wenhao Zhang, Xiaojie Guo, Kang Yang, Ruiyu Zhu, Yu Yu, and Xiao Wang. Efficient actively secure DPF and RAM-based 2PC with one-bit leakage. In *2024 IEEE Symposium on Security and Privacy*, pages 561–577. IEEE Computer Society Press, May 2024.

A More Preliminaries

A.1 Rényi Divergence

We recall the definition of Rényi divergence and its known properties, following the formulation of [BLR+18].

Definition 5 (Rényi Divergence)**.** *For any two discrete probability distributions P and Q such that $Supp(P) \subseteq Supp(Q)$ and $a \in (1, \infty)$, we define the Rényi divergence of order a by*

$$R_a(P||Q) = \left(\sum_{x \in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}$$

For $a = \infty$

$$R_\infty(P||Q) = \max_{x \in Supp(P)} \frac{P(x)}{Q(x)}$$

Lemma 11. *Let P and Q be two discrete probability distributions such that $Supp(P) \subseteq Supp(Q)$.*
Data Processing Inequality *Let $a \in [1, \infty]$. For any function f, where X^f denotes the distribution of $f(x)$ induced by sampling $x \leftarrow X$, we have*

$$R_a(P^f||Q^f) \leq R_a(P||Q).$$

Probability Preservation *Let $E \subseteq Supp(Q)$ be an arbitrary event. Then we have*

$$P(E) \leq Q(E) \cdot R_\infty(P||Q).$$

Multiplicativity *For two families of distributions $\{P_i\}_i$ and $\{Q_i\}_i$, we have:*

$$R_a(\prod_i P_i || \prod_i Q_i) = \prod_i R_a(P_i||Q_i)$$

Lemma 12. ***(Tailcut).*** *(Lemma 2.2 [Pre17]) Let $\mathcal{D}$ be a distribution and $\mathcal{D}_\delta$ its truncation at some threshold t, where $\mathcal{D}(|z| > t) \leq \delta$ for some $\delta > 0$. If $\frac{\mathcal{D}_\delta(x)}{\mathcal{D}(x)} \leq 1 + \delta$ for all $x \in Supp(\mathcal{D}_\delta)$, then for $a \in (1, +\infty]$:*

$$R_a(\mathcal{D}_\delta||\mathcal{D}) \leq 1 + \delta$$

Lemma 13. ***(Relative Error).*** *(Lemma 3 [Pre17]) Let $\mathcal{D}$, $\mathcal{D}_\delta$ be two distributions such that:*

- *$Supp(\mathcal{D}_\delta) = Supp(\mathcal{D})$*
- *$\exists \delta > 0$ such that $1 - \delta \leq \frac{\mathcal{D}_\delta}{\mathcal{D}} \leq 1 + \delta$ over $Supp(\mathcal{D}_\delta)$*

Then, for $a \in (1, +\infty)$:

$$R_a(\mathcal{D}_\delta||\mathcal{D}) \leq \big(1 + \frac{a(a-1)\delta^2}{2(1-\delta)^{a+1}}\big)^{\frac{1}{a-1}} \underset{\delta \to 0}{\sim} 1 + \frac{a\delta^2}{2}$$

Lemma 14. ***(Mean Bound).*** *(Lemma 4.2 [MR04]) Let Λ be an n-dimensional lattice, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^n$ a unit vector, and reals $\epsilon \in (0, 1)$, $\sigma \geq 2\eta_\epsilon(\Lambda)$. Then:*

$$\mathbb{E}_{\mathbf{x} \leftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\langle \mathbf{x} - \mathbf{c}, \mathbf{u} \rangle] \leq \frac{\sqrt{2\pi}\epsilon\sigma}{1-\epsilon}$$

Lemma 15. ***(Tail Bound).*** *(Lemma 4.4 [MR04]) Let Λ be an n-dimensional lattice, $\mathbf{c} \in \mathbb{R}^n$, and reals $\epsilon \in (0, 1)$, $\sigma \geq \eta_\epsilon(\Lambda)$. Then:*

$$\mathbb{P}_{\mathbf{x} \leftarrow D_{\Lambda,\sigma,\mathbf{c}}}[||\mathbf{x} - \mathbf{c}|| \geq \sigma\sqrt{2\pi n}] \leq \frac{1+\epsilon}{1-\epsilon} 2^{-n}$$

Lemma 16. ***(Optimal Rényi Order).*** *(Lemma 13 [FGdG+24]) For $\lambda \in \mathbb{N}$, let E_1, E_2 be two events such that $\Pr[E_1] \geq 2^{-\lambda}$. Assume that for any $Q \in \mathbb{N}$ and $a \in (1, \infty)$ the Rényi divergence between two arbitrary distributions is at most $R_a \in [1, \infty)$, and*

$$\Pr[E_2] \leq R_a^Q \cdot \Pr[E_1]^{\frac{a-1}{a}} \quad \forall a > 1$$

Then,

$$-\log(\Pr[E_2]) \leq -\log(\Pr[E_1]) - \min_{a>1} \big(Q \log R_a + \frac{\lambda}{a}\big)$$

Proposition 1. *(Prop. 4 [HPRR20]). Let $\mathcal{P}$ and $\mathcal{Q}$ denote two distributions of a $N-$tuple of random variables $(x_i)_{i<N}$. For $0 \leq i < N$, assume $\mathcal{P}_i$ (resp $\mathcal{Q}_i$) is a marginal distribution of X_i, and let $\mathcal{P}_{i|<i}(.|x_{<i})$ denote the conditional distribution of x_i given that $(x_0 \dots x_{i-1}) = x_{<i}$. Let $a > 1$. Suppose for all $0 \leq i < N$, there exists $r_{a,i} \geq 1$ such that for all $i-$tuple $x_{<i}$ in the support of $\mathcal{Q}$ restricted to its first i variables,*

$$R_a(\mathcal{Q}_{i|x_{<i}}, \mathcal{P}_{i|x<i}) \leq r_{a,i}$$

Then:

$$R_a(\mathcal{Q}, \mathcal{P}) \leq \prod_{i<N} r_{a,i}$$

Functionality $\mathcal{F}_{\mathsf{ABB}}$

Initialize: On receiving (init) from $\mathcal{P}_i$, send $\Gamma_i \leftarrow \mathbb{F}_p$ and $\Delta_i \leftarrow \mathbb{F}_{2^\lambda}$ to $\mathcal{P}_i$.
Bernoulli: On receiving (Ber, x, y) or $(\mathsf{Ber}, \langle\langle x \rangle\rangle, y)$ from all parties where $x \in \{0,1\}^y$, sample $u \leftarrow \{0,1\}^y$ and compute $b = (u < x)$. Execute $\mathsf{Auth}(b, 2)$ so all parties obtain $\langle\langle b \rangle\rangle$.
Mult: On receiving $(\mathsf{Mult}, [\![a]\!], [\![b]\!])$ from all parties, compute $c = a \cdot b \bmod p$ and execute $\mathsf{Auth}(c, p)$.
Add: On receiving $(\mathsf{Add}, [\![a]\!], [\![b]\!])$ from all parties, compute $c = a + b \bmod p$ and execute $\mathsf{Auth}(c, p)$.
AND: On receiving $(\mathsf{AND}, \langle\langle a \rangle\rangle, \langle\langle b \rangle\rangle)$ from all parties, compute $c = a \wedge b$ and execute $\mathsf{Auth}(c, 2)$.
XOR: On receiving $(\mathsf{XOR}, \langle\langle a \rangle\rangle, \langle\langle b \rangle\rangle)$ from all parties, compute $c = a \oplus b$ and execute $\mathsf{Auth}(c, 2)$.
A2B: On receiving $(\mathsf{A2B}, [\![a]\!])$, reconstruct $a \in \mathbb{F}_p$ and execute $\mathsf{Auth}(a[i], 2)$ for $i \in [\lceil \log p \rceil]$ so parties obtain Boolean shares of a.
B2A: On receiving $(\mathsf{B2A}, \{\langle\langle a[i] \rangle\rangle\}_{i \in [\lceil \log p \rceil]})$, reconstruct a and execute $\mathsf{Auth}(a, p)$.

Figure 9: **Functionality for the MPC black box.**

Macro $\mathsf{Auth}(\mathbf{x}, q, m)$

On input $\mathbf{x}$ and $q \in \{2, p\}$, the subroutine interacts with all parties and generates authenticated shares of x. If $q = 2$, $(\phi_i = \Delta_i)_{i \in [N]}$ and if $q = p$ then $(\phi_i = \Gamma_i)_{i \in [N]}$.

1. Receive $(\hat{\mathbf{x}}_i, \hat{M}_i[\mathbf{x}])_{i \in \mathcal{M}}$ from the adversary $\mathcal{A}$ on behalf of all malicious parties.
2. For each honest party $i \in \mathcal{H}$, sample $\mathbf{x}_i \leftarrow \mathbb{F}_q^m$ and if $q = 2$ $M_i[\mathbf{x}] \leftarrow \mathbb{F}_{2^\lambda}^m$, $M_i[\mathbf{x}] \leftarrow \mathbb{F}_q^m$ such that

$$\mathbf{x} = \Sigma_{i \in \mathcal{H}} \mathbf{x}_i + \Sigma_{i \in \mathcal{M}} \hat{\mathbf{x}}_i$$

$$M[\mathbf{x}] = \Sigma_{i \in \mathcal{H}} M_i[\mathbf{x}] + \Sigma_{i \in \mathcal{M}} \hat{M}_i[\mathbf{x}]$$

3. Send $(\mathbf{x}_i, M_i[\mathbf{x}])$ to $\mathcal{P}_i$ to each $i \in \mathcal{H}$.

Figure 10: **Macro for distributing authenticated shares.**

A.2 More MPC Preliminaries

Figure 9 describes the arithmetic black box functionality $\mathcal{F}_{\mathsf{ABB}}$ and Figure 11 describes the 2-party DPF with one-bit leakage. Figure 13 describes the 2-party malicious security DPF protocol with one-bit leakage from [ZGY+24].

B Additional Sampling Algorithms

In this part, we present the pseudocode of the following sampling algorithms.

- In Algorithm 4, we present `SamplerZ` from the FALCON spec [PFH+20b] for sampling from one-dimentional Gaussian.
- Algorithm 5 describes the fixed-point version of our sampler from Algorithm 3
- In Algorithm 6, we present BerExp(u) sampling from Bernoulli distribution $\mathcal{B}(e^{-u})$.
- In Algorithm 7, we present $\mathrm{Geo}_{2^\ell}(p)$ sampling from Geometric distribution $\mathcal{G}_{2^\ell}(p)$.

Algorithm 4 `SamplerZ` from FALCON

Require: $\mu \in [0,1]$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$.
Ensure: $z \leftarrow \mathcal{N}_{\mathbb{Z}}(\mu, \sigma^2)$

while true **do**
  $y \leftarrow \mathcal{N}_{\mathbb{Z}^+}(0, \sigma_{\max}^2)$
  $b \leftarrow \{0,1\}$
  $z := (2b-1)y + b$
  $x := \frac{y^2}{2\sigma_{\max}^2} - \frac{(z-\mu)^2}{2\sigma^2}$
  if $\mathcal{B}(\frac{\sigma_{\min}}{\sigma} e^x)$ **then**
    Return z

Algorithm 5 Fixed-point version of Algorithm 3

Require: $\mu \in [0,1]$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$. Let $\tau = 0.5$
Ensure: $z \leftarrow \mathcal{N}_M(\mu, \sigma^2)$

while true **do**
  $z_0 \leftarrow \mathsf{Geo}_{M-1}(e^{-\tau})$
  $b \leftarrow \{0,1\}$
  $z = (2b-1)z_0 + b$
  $x = \frac{(z-\mu)^2}{2\sigma^2} - z_0 \cdot \tau + \frac{\sigma_{\max}^2 \cdot \tau^2}{2}$
  if $\mathsf{Ber}(\frac{\sigma_{\min}}{\sigma}) \cdot \mathsf{BerExp}(x)$ **then**
    Return z

Algorithm 6 $\mathsf{BerExp}(u)$ [WYF+23]

Require: $u = \sum_{i=a}^{i=b} 2^i u_i$
Ensure: b sampled from Bernoulli(e^{-u})

$p_i = e^{-2^i}$ for $\forall i \in [a,b]$
for $i \in [a,b]$ **do**
  $b_i = \text{Bernoulli}(p_i)$
$b = \wedge_{i=a}^{i=b}(\neg u_i \vee b_i)$

Algorithm 7 $\mathsf{Geo}_{2^\ell}(p)$ [WYF+23]

Require: Parameter p, range $[0, 2^\ell)$.
Ensure: $x \leftarrow \mathcal{G}_{2^\ell}(p)$.

$p_i = p^{2^i}/(1+p^{2^i})$ for $i \in \{0, \ldots, \ell-1\}$.
for $i \in \{0, \ldots, \ell-1\}$ **do**
  $b_i = \mathcal{B}(p_i)$.
$x = \sum_{i=0}^{\ell-1} 2^i b_i$.

Functionality $\mathcal{F}_{\mathsf{DPF}}$

Gen: Upon receiving $(\mathsf{gen}, \{\omega^{(0)}, \ldots, \omega^{(n)}\})$ from $\mathcal{P}_0$ and (gen) from $\mathcal{P}_1$:

1. Send (init) to $\mathcal{F}_{\mathsf{2PC}}$ (Figure 12) to obtain Δ_0, Δ_1 and compute $\Delta = \Delta_0 \oplus \Delta_1$.
2. Let $\omega = \sum_{i=0}^{n-1} \omega^{(i)} 2^i$ and $N = 2^n$. Compute a unit vector $\mathbf{u} \in \mathbb{F}_2^N$ such that $\mathbf{u}[\omega] = 1$ and $\mathbf{u}[j] = 0$ for $j \neq \omega$. Compute $M[\mathbf{u}] = \mathbf{u} \cdot \Delta$.
3. Sample $\mathbf{u}_0 \leftarrow \mathbb{F}_2^N$, $M_0[\mathbf{u}] \leftarrow \mathbb{F}_{2^\lambda}^N$ and set $\mathbf{u}_1 = \mathbf{u} \oplus \mathbf{u}_0$, $M_1[\mathbf{u}] = M[\mathbf{u}] \oplus M_0[\mathbf{u}]$.
4. Send $(\mathbf{u}_b, M_b[\mathbf{u}])$ to each party $\mathcal{P}_b$.

One-bit leakage challenge: This command is allowed to be called only once. Upon receiving (check) from both parties, do the following:

1. Wait for predicate $P : \mathbb{F}_2^N \times \mathbb{F}_{2^\lambda}^N \to \mathbb{F}_2$.
2. If $P(\mathbf{u}, M[\mathbf{u}]) = 0$ abort.

Figure 11: **Functionality for maliciously secure 2-party DPF** with one-bit leakage.

C Key Generation

Algorithm 8 describes the Gram-Schmidt orthogonalization process used in the Initialize step of $\mathcal{F}_{\mathsf{ThFALCON}}$ (Figure 2).

Algorithm 8 GramSchmidtProcess($\mathbf{B}$) [Pre16]

Require: Basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$
Ensure: Gram-Schmidt reduced basis $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\}$

for $i \leftarrow 1 \ldots n$ **do**
  $\tilde{\mathbf{b}}_\mathbf{i} \leftarrow \mathbf{b}_\mathbf{i}$
  for $j \leftarrow 1, \ldots, i-1$ **do**
    $L_{i,j} \leftarrow \langle \mathbf{b}_i, \tilde{\mathbf{b}}_j \rangle / ||\tilde{\mathbf{b}}_j||^2$
    $\tilde{\mathbf{b}}_i \leftarrow \tilde{\mathbf{b}}_i - L_{i,j} \tilde{\mathbf{b}}_j$
return $\tilde{\mathbf{B}} = \{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_n\}$

D More on Baseline

We instrument the Falcon reference implementation by counting the number of double operations (written as `fpr`) in the code and all the integer operations done as part of `samplerZ`. We make no change in the reference implementation except change the private lazy Bernoulli loop in the `BerExp` function to a fixed 64 bit comparison as MPC does not allow for private control flows. Note that we do not count the operations done to get the inverse FFT of the `ffSampler` output as it can be revealed and thus, the later operations can be done in public. This gives us the number of double precision additions, multiplications, truncations, floor and integer additions, multiplications, and bitwise ANDs. finally in table 5 we report the number of double precision additions, multiplications and the total number of AND gates using the circuit sizes from the respective Bristol format circuits.

Algorithm 9 KeyGen(r)

Require: Randomness r, parameters Φ, q, σ, $\sigma_{\min}$, precision f.
Ensure: Trapdoor basis $\mathbf{B}$, normalised GSO basis $\hat{\mathbf{B}}$, ccs_i and θ_i for $i \in [2n]$, public key h.

$(\bar{f}, \bar{g}, \bar{F}, \bar{G}) \leftarrow \mathsf{NTRUGen}(\Phi, q; r)$
$\mathbf{B} \leftarrow \begin{bmatrix} \bar{g} & -\bar{f} \\ \bar{G} & -\bar{F} \end{bmatrix}$
$\tilde{\mathbf{B}} \leftarrow \mathsf{GramSchmidtProcess}(\mathbf{B})$ (Algorithm 8)
for $i \in [2n]$ **do**
    $\hat{\mathbf{b}}_i \leftarrow \tilde{\mathbf{b}}_i \cdot 2^f / \|\tilde{\mathbf{b}}_i\|^2$
    $\mathsf{ccs}_i \leftarrow (\sigma_{\min}/\sigma) \cdot \|\tilde{\mathbf{b}}_i\| \cdot 2^{f-16}$
    $\theta_i \leftarrow \|\tilde{\mathbf{b}}_i\|^2/(2\sigma^2)$
$h \leftarrow \bar{g} \cdot \bar{f}^{-1} \bmod q$
return $\mathbf{B}$, $\hat{\mathbf{B}} = (\hat{\mathbf{b}}_i)_{i\in[2n]}$, $(\mathsf{ccs}_i)_{i\in[2n]}$, $(\theta_i)_{i\in[2n]}$, h

Functionality $\mathcal{F}_{\mathsf{2PC}}$

Initialize: On receiving (init) from $\mathcal{P}_b$, send $\Delta_b \leftarrow \mathbb{F}_{2^\lambda}$ to $\mathcal{P}_b$.
Auth: On receiving (auth, ω) from $\mathcal{P}_b$ and (auth) from $\mathcal{P}_{1-b}$:

1. Sample $K \leftarrow \{0,1\}^\lambda$ and send it to $\mathcal{P}_b$.
2. Compute $M = K + \omega \cdot \Delta_{1-b}$ and send it to $\mathcal{P}_{1-b}$.

Authenticated Rand: On receiving (aRand) from $\mathcal{P}_0$ and (aRand) from $\mathcal{P}_1$,

1. Sample r_0, $r_1 \leftarrow \mathbb{F}_{2^\lambda}$ and compute $r = r_0 \oplus r_1$.
2. Sample $M_0[r] \leftarrow \mathbb{F}_{2^\lambda}$ and compute $M_1[r] = M_0[r] \oplus r \cdot (\Delta_0 \oplus \Delta_1)$.
3. Send $(r_b, M_b[r])$ to $\mathcal{P}_b$ for $b \in \{0,1\}$.

Unauthenticated Rand: On receiving (rand) from both parties, sample $W \leftarrow \mathbb{F}_{2^\lambda}$ and send it to both parties.
UMult: On receiving $(\mathsf{uMult}, x_b, y_b)$ from $\mathcal{P}_b$ compute $z = (x_0 + x_1) \cdot (y_0 + y_1) \in \mathbb{F}_p$, sample $z_0 \leftarrow \mathbb{F}_p$. Compute $z_1 = z - z_0$ and send z_b to $\mathcal{P}_b$.

Figure 12: **Functionality for two-party computation.**

E Missing Proofs

E.1 Proofs from Section 5

Lemma 17 (Lemma 2, re-stated)**.** *Let $\mathcal{N}_M(\mu, \sigma^2)$ be truncated discrete Gaussian with reals $\mu \in [0,1)$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ and $\mathcal{N}'_M(\mu', \sigma'^2)$ be the output of Algorithm 3 with fixed-point representation such that $|\mu' - \mu| \le 2^{-\tilde{f}}$, $|(1/2\sigma^2)' - (1/2\sigma^2)| \le 2^{-f}$, $|(\frac{\sigma_{\min}}{\sigma}) - (\frac{\sigma_{\min}}{\sigma})'| \le 2^{-f}$. Then,*

$$R_a(\mathcal{N}'_M(\mu', \sigma'^2) || \mathcal{N}_M(\mu, \sigma^2))$$
$$\le 1 + \frac{a(\frac{2^{-\tilde{f}} M}{\sigma_{\min}^2} + (M^2 + M + 1)2^{-f} + \frac{2^{-f}\sigma}{\sigma_{\min}})^2}{2}$$

Proof. For output z from both distributions the acceptance probability is affected, thus, $\frac{\mathcal{N}'_M(z)}{\mathcal{N}_M(z)} = \frac{e^{-x'}(\frac{\sigma_{\min}}{\sigma})'}{e^{-x}\frac{\sigma_{\min}}{\sigma}}$

Protocol Π_{DPF}

Let $\mathcal{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a CCR hash function.
Generate DPF shares: Upon receiving $(\mathsf{gen}, \{\omega^{(0)}, \ldots, \omega^{(n)}\})$ from $\mathcal{P}_0$ and (gen) from $\mathcal{P}_1$:

1. Both parties send (init) to $\mathcal{F}_{\mathsf{2PC}}$ to obtain Δ_b.
2. For $k \in [n]$, $\mathcal{P}_0$ sends $(\mathsf{auth}, \omega^{(k)})$ and $\mathcal{P}_1$ sends (auth) to $\mathcal{F}_{\mathsf{2PC}}$ to obtain $K[\omega^{(k)}]$ and $M[\omega^{(k)}]$ respectively.
3. Both parties send (rand) to $\mathcal{F}_{\mathsf{2PC}}$ to sample a public randomness $W \in \mathbb{F}_{2^\lambda}$. Each party $\mathcal{P}_b$ sets $(s_b^{(0,0)} || t_b^{(0,0)}) := \Delta_b \oplus W \in \{0,1\}^\lambda$.
4. For each $k \in [0, n)$, each party $\mathcal{P}_b$ computes $\mathsf{CW}_b^{(k)}$ as follows:

$$\mathsf{CW}_0^{(k)} := (\bigoplus_{l \in [0,2^k)} \mathcal{H}(s_0^{(k,l)} || t_0^{(k,l)})) \oplus \Delta_0 \oplus M[\omega^{(k)}] \in \{0,1\}^\lambda$$

$$\mathsf{CW}_1^{(k)} := (\bigoplus_{l \in [0,2^k)} \mathcal{H}(s_1^{(k,l)} || t_1^{(k,l)})) \oplus \Delta_1 \oplus (K[\omega^{(k)}] \oplus \omega^{(k)} \cdot \Delta_1) \in \{0,1\}^\lambda$$

   $\mathcal{P}_b$ sends $\mathsf{CW}_b^{(k)}$ to $\mathcal{P}_{1-b}$. Both parties compute $\mathsf{CW}^{(k)} := \mathsf{CW}_0^{(k)} \oplus \mathsf{CW}_1^{(k)}$. For each $l \in [0, 2^k)$ each party $\mathcal{P}_b$ computes:

$$(s_b^{(k+1,2l)} || t_b^{(k+1,2l)}) = \mathcal{H}(s_b^{(k,l)} || t_b^{(k,l)}) \oplus t_b^{(k,l)} \cdot \mathsf{CW}^{(k)}$$

$$(s_b^{(k+1,2l+1)} || t_b^{(k+1,2l+1)}) = \mathcal{H}(s_b^{(k,l)} || t_b^{(k,l)}) \oplus (s_b^{(k,l)} || t_b^{(k,l)}) \oplus t_b^{(k,l)} \cdot \mathsf{CW}^{(k)}$$

5. Both parties $\mathcal{P}_b$ compute $\mathbf{u}_b[l] = t_b^{(n,l)}$ and $M_b[\mathbf{u}[l]] = (s_b^{(n,l)} || t_b^{(n,l)})$ for each $l \in [0, 2^n)$.
6. Both parties $\mathcal{P}_b$ send (aRand) to $\mathcal{F}_{\mathsf{2PC}}$ to obtain $(r_b, M_b[r])$. Then both parties send (rand) to $\mathcal{F}_{\mathsf{2PC}}$ to sample a random challenge $\chi \in \mathbb{F}_{2^\lambda}$ and compute

$$a_b := \sum_{l \in [0,2^n)} \chi^l \cdot \mathbf{u}_b[l] + r_b$$

$$M_b[a] := \sum_{l \in [0,2^n)} \chi^l \cdot M_b[\mathbf{u}[l]] + M_b[r]$$

7. Both parties $\mathcal{P}_b$ send a_b to $\mathcal{P}_{1-b}$ and compute $a = a_0 + a_1 \in \mathbb{F}_{2^\lambda}$. Both parties compute $V_b := M_b[a] + a \cdot \Delta_b$ and calls $\mathcal{F}_{\mathsf{com}}$ to commit to V_b.
8. Both parties call $\mathcal{F}_{\mathsf{com}}$ to open V_b. Then, both parties check $V_0 = V_1$ and abort if the check fails.

   Output: Each party $\mathcal{P}_b$ obtains $(\mathbf{u}_b, M_b[\mathbf{u}])$.

Figure 13: **2-party malicious DPF protocol with one-bit leakage.** [ZGY+24]

Let us define $\delta_x := |x - x'|$, $\delta_{ccs} := |(\frac{\sigma_{\min}}{\sigma}) - (\frac{\sigma_{\min}}{\sigma})'|$. For $\tau = 0.5$, we have $\delta_\tau \leq 2^{-f}$. Below we focus on obtaining δ_x.

$$x = \frac{(z-\mu)^2}{2\sigma^2} - z_0 \cdot \tau + \frac{\sigma_{\max}^2 \cdot \tau^2}{2}$$

Given $\sigma_{\max}$ is public the last term can be pre-computed up to desired precision. For simplicity,

let us say $\delta_{x3} = |\sigma^2_{\max}\tau/2 - \sigma'^2_{\max}\tau'/2| \le 2^{-f}$. Thus,

$$\delta_x = \Big|\frac{(z-\mu)^2}{2\sigma^2} - \frac{(z-\mu')^2}{2\sigma'^2} - z_0 \cdot (\tau - \tau') + \delta_{x3}\Big|$$

Let $b = 1/2\sigma^2$ and $b' = 1/2\sigma'^2$.

$$\delta_x = |(z-\mu)^2 b - (z-\mu)^2 b' - z_0(\tau - \tau') + \delta_{x3}|$$

Now consider fixed-point errors in each term: $\mu - \mu' = \delta_\mu$, $b - b' = \delta_b$ and $\tau - \tau' = \delta_\tau$. We expand around (μ, b, τ) and collect error terms. Thus,

$$\delta_x = \big|(z-\mu)^2 b - (z-\mu+\delta_\mu)^2(b-\delta_b) - z_0\delta_\tau + \delta_{x3}\big|$$

$$\delta_x = \big| - \delta_\mu b(2z - 2\mu - \delta_\mu) + (z-\mu+\delta_\mu)^2\delta_b - z_0\delta_\tau + \delta_{x3}\big|$$

We know that $\delta_\mu \le 2^{-\tilde{f}}$, $\delta_b \le 2^{-f}$, $\delta_\tau \le 2^{-f}$ and $\delta_{x3} \le 2^{-f}$. Thus,

$$\delta_x \le \big| - 2^{-\tilde{f}} b(2z - 2\mu) + (z-\mu)^2 2^{-f} + 2^{-f} + z_0 2^{-f}\big|$$

By definition $b_{\max} = 1/2\sigma^2_{\min}$, $|z-\mu|_{\max} = M$. For simplicity let $z_{0\max}$ be M. Thus,

$$\delta_x \le \frac{2^{-\tilde{f}} M}{\sigma^2_{\min}} + M^2 2^{-f} + M 2^{-f} + 2^{-f}$$

Thus,

$$e^{-\delta_x}(1 - \frac{\delta_{ccs}}{\frac{\sigma_{\min}}{\sigma}}) \le \frac{\mathcal{N}'_M(z)}{\mathcal{N}_M(z)} \le e^{\delta_x}(1 + \frac{\delta_{ccs}}{\frac{\sigma_{\min}}{\sigma}})$$

$$(1-\delta_x)(1 - \frac{\delta_{ccs}}{\frac{\sigma_{\min}}{\sigma}}) \le \frac{\mathcal{N}'_M(z)}{\mathcal{N}_M(z)} \le (1+\delta_x)(1 + \frac{\delta_{ccs}}{\frac{\sigma_{\min}}{\sigma}})$$

$$(1 - \delta_x - \frac{\delta_{ccs}\sigma}{\sigma_{\min}}) \le \frac{\mathcal{N}'_M(z)}{\mathcal{N}_M(z)} \le (1 + \delta_x + \frac{\delta_{ccs}\sigma}{\sigma_{\min}})$$

By Lemma 13,

$R_a(\mathcal{N}'_M(\mu', \sigma'^2)||\mathcal{N}_M(\mu, \sigma^2)) \le 1 + \frac{a(\delta_x + \frac{2^{-f}\sigma}{\sigma_{\min}})^2}{2}$ □

E.2 Proofs from Section 6

Lemma 18 (Lemma 5, re-stated)**.** π_{Geo} *(Figure 4b) outputs samples from the same distribution as Algorithm 7.*

Proof. The procedure computes the public probabilities $p_i = q^{2^i}/(1+q^{2^i})$ for $i \in [\ell]$ and uses $\mathcal{F}_{\mathsf{ABB}}$ to sample $x[i] \leftarrow \mathcal{B}(p_i)$. This is exactly the bit-by-bit sampling performed by Algorithm 7, so both outputs follow the same distribution. □

Lemma 19 (Lemma 6, re-stated)**.** π_{Ber} *(Figure 4a) on input* $\langle\langle u \rangle\rangle_{a.b}$ *outputs samples from* $\mathcal{B}(e^{-u})$ *matching Algorithm 6.*

Proof. The procedure decomposes $e^{-u} = \prod_{j\in[a+b]} e^{-2^{j-b}u[j]}$ and samples $x_j \leftarrow \mathcal{B}(e^{-2^{j-b}})$ for each bit position. The output $y = \bigwedge_j(\neg u[j] \vee x_j)$ equals 1 with probability e^{-u}, matching Algorithm 6. Since $\delta_x \le 2^{-42}$ from the precision analysis, the fixed-point range $[-42, 13]$ suffices. □

Lemma 20 (Lemma 7, re-stated)**.** π_{Gauss} *(Figure 4c) outputs samples from* $\mathcal{N}_M(\mu, \sigma^2)$ *matching Algorithm 5.*

Proof. The procedure obtains z_0 from π_{Geo} (correct by Lemma 5) and computes the signed sample $z = (2y-1)z_0 + y$. The rejection value x is computed via a Boolean circuit that evaluates $(z-u)^2$ up to f' bits, multiplies by θ_i up to f bits, then shifts by $f+f'$ bits and discards the lowest $f'-42$ bits. This matches the fixed-point arithmetic in Algorithm 5. The acceptance test uses π_{Ber} (correct by Lemma 6) to sample from $\mathcal{B}(e^{-x})$ and $\mathcal{B}(\mathsf{ccs}_i)$. Thus, the output distribution matches Algorithm 5. □

Lemma 21 (Lemma 8, re-stated)**.** π_{Klein} *(Figure 5) correctly instantiates Algorithm 2 using* $\mathcal{F}_{\mathsf{ABB}}$*.*

Proof. The inner product $d_i = \langle \mathbf{t}_i, \hat{\mathbf{b}}_i \rangle$ is computed in $\mathbb{F}_p$. Since $p > 2^{16+f+1+\log n}$, no wraparound occurs, so converting to Boolean shares $\langle\langle d_i \rangle\rangle_{16.f}$ preserves exactness. Both the functionality and procedure discard the $\log n + 1$ least significant bits before invoking the Gaussian sampler. By Lemma 7, π_{Gauss} produces correct samples. The output z is converted back to arithmetic shares, and the products $z \cdot \mathbf{b}_i$ are computed in $\mathbb{F}_p$ without wraparound. The local updates $[\![\mathbf{t}_{i-1}]\!]$ and $[\![\mathbf{v}_{i-1}]\!]$ match the ideal computation. □

Theorem 4 (Theorem 2, re-stated)**.** *Protocol* Π_{sign} *(Figure 6) UC-realizes* $\mathcal{F}_{\mathsf{ThFALCON}}$ *(Figure 2) in the* $\mathcal{F}_{\mathsf{ABB}}$*-hybrid.*

Proof. Correctness: The Initialize step of Π_{sign} invokes $\mathcal{F}_{\mathsf{ABB}}$ to execute the same key generation computation as the Initialize command of $\mathcal{F}_{\mathsf{ThFALCON}}$, so all parties obtain the correct authenticated shares of $\mathbf{B}$, $\hat{\mathbf{B}}$, and ccs, along with the public key h. For the Sign command, by Lemma 8, π_{Klein} produces samples from the same distribution as Algorithm 2. All other operations (hashing, signature computation, norm check) are identical in $\mathcal{F}_{\mathsf{ThFALCON}}$ and Π_{sign}.

Security: All computation on private inputs in both the Initialize and Sign steps is performed via $\mathcal{F}_{\mathsf{ABB}}$, and no additional messages are exchanged outside of $\mathcal{F}_{\mathsf{ABB}}$. The simulator uses the $\mathcal{F}_{\mathsf{ABB}}$ simulator to handle both steps, and the simulation is straightforward and omitted. □

E.3 Proofs from Section 7

Lemma 22 (Lemma 9, re-stated)**.** *If* $\mathbf{z} \neq \mathbf{x} \cdot y$*, then the batch-check in step 5 of* Π_{aVOLE} *(verifying* $M[\mathbf{x}] = \Gamma \cdot \mathbf{x}$ *or* $M[\mathbf{z}] = \Gamma \cdot \mathbf{z}$*) fails with all but negligible probability.*

Proof. For $j \in \mathcal{M}$, let $\boldsymbol{\omega}_j^i$, $\boldsymbol{\beta}_j^i$, Γ_j^i, y_j^i, $M_j^i[y]$ be the values it uses in step 2. Let ω_j^*, $\boldsymbol{\beta}_j^*$, $\mathsf{CW}^{(i,j)*}$, $\mathbf{b}^{(i,j)*}$ be the honest values. Let the error added by malicious parties in step 4 (after LPN) be $\tau_{M[x]}$, $\tau_{\mathbf{z}}$ and $\tau_{M[z]}$.

Thus,

$$\delta_{M[\mathbf{x}]} = H \cdot \sum_{i \in [N]} (\mathbf{b}_i - \mathbf{b}_i^*) + \tau_{M[x]}$$

For $i \in \mathcal{H}$,

$$\mathbf{b}_i - \mathbf{b}_i^* = \sum_{j \in \mathcal{M}} (\mathbf{b}_i^{(i,j)} + \mathbf{b}_i^{(j,i)}) - (\mathbf{b}_i^{(i,j)*} + \mathbf{b}_i^{(j,i)*})$$

Let $\delta_{\mathbf{b}_i}$, $\delta_{\mathbf{c}_i}$ and $\delta_{\mathbf{d}_i}$ be the errors introduced by the malicious parties in step 3, thus for $i \in \mathcal{M}$

$$\mathbf{b}_i - \mathbf{b}_i^* = \sum_{j \in \mathcal{H}} (\mathbf{b}_i^{(i,j)} + \mathbf{b}_i^{(j,i)}) - (\mathbf{b}_i^{(i,j)*} + \mathbf{b}_i^{(j,i)*}) + \delta_{b_i}$$

Thus,

$$\mathbf{b} - \mathbf{b}^* = \sum_{i\in\mathcal{M}}(\mathbf{b}_i - \mathbf{b}_i^*) + \sum_{i\in\mathcal{H}}(\mathbf{b}_i - \mathbf{b}_i^*)$$

$$\sum_{i\in\mathcal{H},j\in\mathcal{M}}(\mathbf{b}^{(i,j)} + \mathbf{b}^{(j,i)} - \mathbf{b}^{(i,j)*} - \mathbf{b}^{(j,i)*}) + \sum_{j\in\mathcal{M}}\delta_{b_j}$$

$$= \sum_{i\in\mathcal{H},j\in\mathcal{M}}\big(\mathbf{u}_i(\Gamma_j^i - \Gamma_j) + (\mathbf{u}_j^{(i,0)} - \mathbf{u}_j)\Gamma_i\big) + \sum_{j\in\mathcal{M}}\delta_{b_j}$$

Similarly,

$$\mathbf{c} - \mathbf{c}^* = \sum_{i\in\mathcal{H},j\in\mathcal{M}}\big(\mathbf{u}_i(y_j^i - y_j) + (\mathbf{u}_j^{(i,1)} - \mathbf{u}_j)y_i\big) + \sum_{j\in\mathcal{M}}\delta_{c_j}$$

$$\mathbf{d} - \mathbf{d}^* = \sum_{i\in\mathcal{H},j\in\mathcal{M}}\big(\mathbf{u}_i(M_j^i[y] - M_j[y]) + (\mathbf{u}_j^{(i,2)} - \mathbf{u}_j)M_i[y]\big) + \sum_{j\in\mathcal{M}}\delta_{d_j}$$

The batch-check for $\mathbf{x}$ fails if

$$\delta_x\Gamma \neq H\left(\sum_{i\in\mathcal{H},j\in\mathcal{M}}\big(\mathbf{u}_i(\Gamma_j^i - \Gamma_j) + (\mathbf{u}_j^{(i,0)} - \mathbf{u}_j)\Gamma_i\big) + \sum_{j\in\mathcal{M}}\delta_{b_j}\right) + \tau_{M[x]}$$

Given $\Gamma \in \mathbb{F}_p$ is unknown to the adversary $\mathcal{A}$, for non-zero δ_x, $\Gamma_j^i - \Gamma_j$, $\mathbf{u}_j^{i,0} - \mathbf{u}_j$, δ_{b_j} or $\tau_{M[x]}$ the above check fails with overwhelming probability. Similarly, we can say the batch-check for $\mathbf{z}$ fails for non-zero $\mathbf{u}_j^{i,1} - \mathbf{u}_j$, $\mathbf{u}_j^{i,2} - \mathbf{u}_j$, $y_j^i - y_j$, $M_j^i[y] - M_j[y]$, δ_{c_j}, τ_z or $\tau_{M[z]}$.

□

Lemma 23 (Lemma 10, re-stated)**.** *If all parties are honest, then* Π_{aVOLE} *(Figure 8) generates the correct correlation specified by command* (aVOLE) *in* $\mathcal{F}_{\mathsf{aVOLE}}$ *(Figure 7).*

Proof. Let us prove the correlation on $\mathbf{u}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$.

$$\sum_{i\in[N]}\mathbf{b}_i = \sum_{i\in[N]}\Big(\sum_{j\neq i}(\mathbf{b}_i^{(i,j)} + \mathbf{b}_i^{(j,i)}) + \mathbf{u}_i\cdot\Gamma_i\Big)$$

$$= \sum_{i\in[N]}\sum_{j\neq i}(\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)}) + \sum_{i\in[N]}\mathbf{u}_i\cdot\Gamma_i$$

Let us focus on $\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)}$ as obtained in step 2, with $\mathcal{P}_i$ and $\mathcal{P}_j$.
For each $k \in [t]$ we can obtain the following relation,

$$(\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)})[\frac{k\mathbb{N}}{t} + l] =$$

$$-\mathcal{H}(\mathbf{w}_i^{(i,j)}[l]) - \mathbf{v}_i^{(i,j)}[l]\cdot\mathsf{CW}^{(i,j)}[0] + \mathcal{H}(\mathbf{w}_j^{(i,j)}[l]) + \mathbf{v}_j^{(i,j)}[l]\cdot\mathsf{CW}^{(i,j)}[0]$$

$$= (\mathcal{H}(\mathbf{w}_j^{(i,j)}[l]) - \mathcal{H}(\mathbf{w}_i^{(i,j)}[l])) + (\mathbf{v}_j^{(i,j)}[l] - \mathbf{v}_i^{(i,j)}[l])\cdot\mathsf{CW}^{(i,j)}[0]$$

For $l \in [N/t]$, such that $l \neq \boldsymbol{\omega}_i[k]$, $\mathbf{b}_i^{(i,j)}[\frac{k\mathbb{N}}{t} + l] + \mathbf{b}_j^{(i,j)}[\frac{k\mathbb{N}}{t} + l] = 0$. For step 2d. we have the following correction word for k,

$$\mathsf{CW}^{(i,j)}[0] = (\mathbf{v}_j^{(i,j)}[\boldsymbol{\omega}_i[k]] - \mathbf{v}_i^{(i,j)}[\boldsymbol{\omega}_i[k]])$$

$$\cdot \left(\mathcal{H}(\mathbf{w}_i^{(i,j)}[\boldsymbol{\omega}_i[k]]) - \mathcal{H}(\mathbf{w}_j^{(i,j)}[\boldsymbol{\omega}_i[k]]) + \boldsymbol{\beta}_i[k]\Gamma_j\right)$$

Plugging this to get $\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)}$,

$$(\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)})[\frac{k\mathbb{N}}{t} + \boldsymbol{\omega}_i[k]] = \boldsymbol{\beta}_i[k] \cdot \Gamma_j$$

Thus, $\mathbf{b}_i^{(i,j)} + \mathbf{b}_j^{(i,j)} = \mathbf{u}_i \cdot \Gamma_j$. Similarly, we can get $\mathbf{c}_i^{(i,j)} + \mathbf{c}_j^{(i,j)} = \mathbf{u}_i \cdot y_j$ and $\mathbf{d}_i^{(i,j)} + \mathbf{d}_j^{(i,j)} = \mathbf{u}_i \cdot M_j[y]$.

Thus,

$$\mathbf{b} = \sum_{i\in[N]} \Big(\sum_{j\neq i} \mathbf{u}_i \cdot \Gamma_j + \mathbf{u}_i \cdot \Gamma_i \Big) = \sum_{i\in[N]} \mathbf{u}_i \cdot \sum_{j\in[N]} \Gamma_j$$

Similarly, we have $\mathbf{c} = \mathbf{u} \cdot y$ and $\mathbf{d} = \mathbf{u} \cdot M[y] = \mathbf{u} \cdot y \cdot \Gamma = \mathbf{c} \cdot \Gamma$.

$\mathbf{x}$, $\mathbf{z}$, $M[\mathbf{x}]$ and $M[\mathbf{z}]$ are the LPN outputs using $\mathbf{u}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ respectively. Since we maintain all linear correlations after applying the parity check matrix, we have $\mathbf{z} = \mathbf{x} \cdot y$, $M[\mathbf{x}] = \mathbf{x} \cdot \Gamma$, and $M[\mathbf{z}] = \mathbf{z} \cdot \Gamma$.

□

Theorem 5 (Theorem 3, re-stated)**.** *Assuming decisional dual-LPN is hard with static leakage, protocol* Π_{aVOLE} *(Figure 8) securely instantiates command* (aVOLE) *of* $\mathcal{F}_{\mathsf{ABB}}$ *in the* $(\mathcal{F}_{\mathsf{DPF}}, \mathcal{F}_{\mathsf{2PC}}, \mathcal{H})$*-hybrid model, where* $\mathcal{H} : \mathbb{F}_{2^\lambda} \to \mathbb{F}_p$ *is a random oracle.*

Proof. We construct a simulator $\mathcal{S}$ as follows:

2a. On behalf of $\mathcal{F}_{\mathsf{DPF}}$, for each $i \in \mathcal{H}$ and $j \in \mathcal{M}$, $\mathcal{S}$ receives $\boldsymbol{\omega}_j^i[k]$ from the malicious party $\mathcal{P}_j$ and sends $\mathbf{v}_j^{(i,j)}, \mathbf{v}_j^{(j,i)} \leftarrow \{0,1\}^{\mathbb{N}/t}$ and $\mathbf{w}_j^{(i,j)}, \mathbf{w}_j^{(j,i)} \leftarrow \mathbb{F}_{2^\lambda}^{\mathbb{N}/t}$ to $\mathcal{P}_j$. For any $j \in \mathcal{M}$, if $\boldsymbol{\omega}_i^j$ is not consistent for all $i \in \mathcal{H}$, $\mathcal{S}$ follows the protocol and aborts in step 5; otherwise $\mathcal{S}$ records all inputs and outputs to $\mathcal{P}_j$.

2b. On behalf of $\mathcal{F}_{\mathsf{2PC}}$, $\mathcal{S}$ receives Γ_j^i, y_j^i, $M_j^i[y]$, and $\boldsymbol{\beta}_j^{i,l}$ for each $l \in [3]$ from $\mathcal{P}_j$ for all $j \in \mathcal{M}$ and $i \in \mathcal{H}$. $\mathcal{S}$ stores all the malicious parties' inputs and sends uniformly random values $\alpha[l]_j^{(i,j)}, \alpha[l]_j^{(j,i)} \leftarrow \mathbb{Z}_p$ for $l \in [3]$ to $\mathcal{P}_j$. If for any $j \in \mathcal{M}$ one of the following checks fails, $\mathcal{S}$ aborts in step 5:

- $\Gamma_j^{i_1} \neq \Gamma_j^{i_2}$ for any $i_1, i_2 \in \mathcal{H}$.
- $y_j^{i_1} \neq y_j^{i_2}$ for any $i_1, i_2 \in \mathcal{H}$.
- $M_j^{i_1}[y] \neq M_j^{i_2}[y]$ for any $i_1, i_2 \in \mathcal{H}$.
- $\boldsymbol{\beta}_j^{(i,0)} \neq \boldsymbol{\beta}_j^{(i,1)}$ or $\boldsymbol{\beta}_j^{(i,1)} \neq \boldsymbol{\beta}_j^{(i,2)}$ for any $i \in \mathcal{H}$.
- $\boldsymbol{\beta}_j^{(i_1,\cdot)} \neq \boldsymbol{\beta}_j^{(i_2,\cdot)}$ for any $i_1, i_2 \in \mathcal{H}$.

2d. For each $i \in \mathcal{H}$ and $j \in \mathcal{M}$, on behalf of $\mathcal{F}_{\mathsf{2PC}}$, $\mathcal{S}$ receives $(\mathsf{A}_j^{(i,j)}, \mathsf{B}_j^{(i,j)} + \alpha_j^{(i,j)}[l])_{l\in[3]}$ and $(\mathsf{A}_j^{(j,i)}, \mathsf{B}_j^{(j,i)} + \alpha_j^{(j,i)}[l])_{l\in[3]}$. $\mathcal{S}$ checks if the values received are consistent with those computed using the DPF outputs stored in step 2a and α shares from step 2b. $\mathcal{S}$ aborts if the malicious parties' input are not correct in step 5. $\mathcal{S}$ sends uniformly random $\mathsf{CW}[l]_j^{(i,j)}$, $\mathsf{CW}[l]_j^{(j,i)} \leftarrow \mathbb{Z}_p$ to the party $\mathcal{P}_j$.

2e. For each $j \in \mathcal{M}$, $\mathcal{S}$ receives the malicious party's share of the correction word and aborts in step 5 if it is different from the recorded value in step 2d. $\mathcal{S}$ sends uniformly random $\mathsf{CW}[l]_i^{(i,j)}, \mathsf{CW}[l]_i^{(j,i)} \leftarrow \mathbb{Z}_p$ to $\mathcal{P}_j$ on behalf of the honest parties $\mathcal{P}_i$ for each $i \in \mathcal{H}$.

4. $\mathcal{S}$ computes $(\mathbf{x}_i, M_i[\mathbf{x}], \mathbf{z}_i, M_i[\mathbf{z}])_{i\in\mathcal{M}}$ as they were computed honestly using the recorded values and observes the malicious parties outputs, $(\hat{\mathbf{x}}_i, \hat{M}_i[\mathbf{x}], \hat{\mathbf{z}}_i, \hat{M}_i[\mathbf{z}])_{i\in\mathcal{M}}$. $\mathcal{S}$ aborts if $\sum_{i\in\mathcal{M}} \mathbf{x}_i \neq \sum_{i\in\mathcal{M}} \hat{\mathbf{x}}_i$, $\sum_{i\in\mathcal{M}} \mathbf{z}_i \neq \sum_{i\in\mathcal{M}} \hat{\mathbf{z}}_i$, $\sum_{i\in\mathcal{M}} M_i[\mathbf{x}] \neq \sum_{i\in\mathcal{M}} \hat{M}_i[\mathbf{x}]$, or $\sum_{i\in\mathcal{M}} M_i[\mathbf{z}] \neq \sum_{i\in\mathcal{M}} \hat{M}_i[\mathbf{z}]$; Otherwise $\mathcal{S}$ sends $(\mathsf{aVOLE}, [\![y]\!]_j, m)$ to $\mathcal{F}_{\mathsf{ABB}}$ on behalf of all malicious parties $j \in \mathcal{M}$.

Hybrid$_0$. $\mathcal{S}$ plays the role of honest parties $\mathcal{P}_i$ for $i \in \mathcal{H}$ with actual inputs $\{y_i,\ M_i[y]\}_{\{i\in\mathcal{H}\}}$, i.e., it is exactly the real-world view.

Hybrid$_1$. Same as **Hybrid**$_0$, except in step 4, $\mathcal{S}$ observes the malicious parties' outputs $(\hat{\mathbf{x}}_i, \hat{M}_i[\mathbf{x}], \hat{\mathbf{z}}_i, \hat{M}_i[\mathbf{z}])_{i\in\mathcal{M}}$ and if any honest party aborts it outputs whatever $\mathcal{A}$ outputs. Otherwise, instead of using LPN to obtain the honest parties outputs $\mathcal{S}$ uses uniformly random values $(\mathbf{x}_i, M_i[\mathbf{x}], \mathbf{z}_i, M_i[\mathbf{z}])_{i\in\mathcal{H}}$ such that $\mathbf{z} = \mathbf{x}\cdot y$, $M[\mathbf{x}] = \Gamma\cdot\mathbf{x}$, and $M[\mathbf{z}] = \Gamma\cdot\mathbf{z}$. The view from **Hybrid**$_0$ and **Hybrid**$_1$ are indistinguishable due to the dual-LPN assumption (Theorem 4) as in **Hybrid**$_0$ $\mathbf{u}_i$, $\mathbf{b}_i$, $\mathbf{c}_i$, and $\mathbf{d}_i$ are uniformly random t-regular vectors.

Hybrid$_2$. Same as **Hybrid**$_1$, except in step 2, instead of using the ideal functionalities $\mathcal{F}_{\mathsf{2PC}}$ (2b and 2d) and $\mathcal{F}_{\mathsf{DPF}}$ (2a) it uses uniformly random values as shown in the simulator construction. The view from **Hybrid**$_2$ is indistinguishable from **Hybrid**$_1$ as: In step 2a, $\mathcal{A}$ obtains random vectors from $\mathcal{F}_{\mathsf{DPF}}$ in both ideal-world and real-world (we remark the advantage of the adversary due to the leakage from DPF later). In step 2b and 2d the adversary obtains its share of the corresponding product, thus, the random value sent in the ideal-world is indistinguishable from the real-world. Note that all three correction words are independent and random due to the use of random oracle $\mathcal{H}$.

Remark on static-leakage in DPF functionality: The probability of leakage of b bits in $\mathcal{F}_{\mathsf{DPF}}$ is 2^{-b}, i.e., it is same as the adversary guessing the value and continuing the protocol without querying the functionality $\mathcal{F}_{\mathsf{DPF}}$ with the predicate.

Hybrid$_3$. Same as **Hybrid**$_2$, except $\mathcal{S}$ aborts as described in the construction above, i.e., the ideal world. To show that the views from **Hybrid**$_2$ and **Hybrid**$_3$ are indistinguishable we show that the simulator $\mathcal{S}$ aborts with the same probability as the honest parties in the real world.

From the unforgeability Lemma 9, we know that the real-world protocol aborts with all but negligible probability if the malicious parties cheat in any step, i.e., for any non-zero values of $\Gamma^i_j - \Gamma_j$, $\mathbf{u}^{i,0}_j - \mathbf{u}_j$, $\mathbf{u}^{i,1}_j - \mathbf{u}_j$, $\mathbf{u}^{i,2}_j - \mathbf{u}_j$, $y^i_j - y_j$, and $M^i_j[y] - M_j[y]$, or inconsistency in the LPN output. Thus, the real-world aborts with overwhelming probability if the simulator aborts.

Finally, by Lemma 10, we know that the real-world shares satisfy the same relation as the ideal-world ($\mathbf{z} = \mathbf{x}\cdot y$) in case of no-abort.

□