MASKAGLIA: A New, Efficient Approach to Masked Discrete Gaussian Sampling

Calvin Abou Haidar[1], Thomas Espitau[2],
Clément Hoffmann[1], and Mehdi Tibouchi[1]

[1] NTT Social Informatics Laboratories, Tokyo, Japan
{calvin.haidar, clement.hoffmann, mehdi.tibouchi}@ntt.com
[2] PQShield
thomas.espitau@pqshield.com

Abstract. Discrete Gaussian sampling is an important operation at the core of many lattice-based cryptosystems, which presents significant challenges from an implementation standpoint. In particular, it is difficult to protect against side-channel attacks. Extensive research has gone into the problem of addressing *timing* side-channel attacks, and as result, *constant-time* discrete Gaussian sampling is now well-understood. However, few papers so far have attempted to achieve protection against stronger side-channel attacks like correlation power analysis via, e.g., masking, and those that have tend to suffer from underwhelming performance.

Focusing on the case of discrete Gaussians with fixed center and standard deviation, the state-of-the-art approach to applying a masking countermeasure is to start from a constant-time cumulative distribution table-based (CDT-based) sampler, possibly with a search tree twist. Such a CDT-based sampler compares a uniform random value to each element of the CDT of the target distribution. Replacing these comparisons with a masked comparison circuit (typically based on a carry-save adder like Kogge–Stone) yields the desired, albeit costly, countermeasure.

In this paper, we propose a very different approach to masked discrete Gaussian sampling. We start from a new rejection-based discrete Gaussian sampler, obtained by discretizing a sampler for the continuous normal distribution related to an algorithm of Marsaglia (1963). We show that our new sampler can be expressed elegantly in terms of uniform and geometric distributions, in a way that is surprisingly friendly to masking, particularly when using bitslicing. The resulting masked, t-probing secure gadget dramatically outperforms previous work. When applied to NIST candidate signature HAWK, we find it to need less than 5 masked AND gates per generated sample on a 32-bit architecture, and about 20 times fewer than the state-of-the-art, comparison tree-based masked sampler of Eid et al. (TCHES 2026). Furthermore, we show that while Eid et al.'s sampler can be sped up with significant tweaks (bitslicing, faster masked comparisons, etc.), the modified gadget still requires 4 to 5 times as many masked AND gates as our techniques.

Keywords: Discrete Gaussian sampling · Masking · Side-channel security · t-probing model · HAWK digital signature

1 Introduction

1.1 Implementation security of discrete Gaussian sampling

As the transition towards post-quantum cryptography (PQC) proceeds forward, the security of PQC implementations, including against side-channel analysis, stands out as a particularly important and urgent problem to tackle. Indeed, proposed attacks abound, and for a number of the "new" operations that were not present in classical cryptography and that PQC schemes rely on, suitable countermeasures have yet to appear, or may suffer from excessive overhead when they exist at all.

A case in point is that of discrete Gaussian sampling. That functionality occurs in many lattice-based cryptosystems, either as a fundamental ingredient of the construction itself (typically in hash-and-sign signature [GPV08, DLP14, PFH+22, CPS+20, BBD+24] and more advanced primitives relying on GPV-style trapdoor sampling) or in order to achieve better parameters than those possible with simpler distributions (including in many homomorphic or threshold schemes, or in signatures like Bliss [DDLL13] or Haetae [CCD+24] that rely on distributions *derived* from discrete Gaussians). Unfortunately, discrete Gaussian sampling is not easy to implement correctly and securely, and has in fact been the target of numerous side-channel and fault attacks so far [BHLY16, PBY17, EFGT17, FKT+20, KA21, MWG+22, GMRR22, WGS22, MKK+23, ZLYW23, GR24, LZY+25, HPT25] including some [Pre23, SD25] against "provably protected" implementations [EFG+22]. The challenge posed by Gaussian sampling implementations is actually cited as a main reason why a scheme like Dilithium [LDK+22] (now ML-DSA) avoids Gaussian distributions altogether, at the cost of somewhat suboptimal parameters. Similarly, Raccoon [dPKPR24] uses sums of uniforms instead of Gaussians specifically for their masking-friendliness.

Early implementations of discrete Gaussian sampling were not *constant-time* (also know as "isochronous"), and could thus fall prey to *timing* side-channels and other single-trace attacks. However, thanks to a long series of works [MW17, BBE+19, ZSS20b, ZSS20a, HPRR20, DFW20], the issue has largely been resolved, and the problem of implementing discrete Gaussian sampling in constant time is now well-understood.

On the other hand, research on the problem of protecting against stronger side-channel attacks like correlation power analysis, using countermeasures like masking, is much more sparse, even in the simple case of discrete Gaussians with fixed, public centers and standard deviations.

For that case (which is the main point of focus of this paper as well), essentially all approaches described in the literature so far are table-based. They start from a constant-time cumulative distribution table-based (CDT-based) sampler (or a variant like Falcon's RCDT base sampler), which compares a uniform random value to each element of the CDT of the target distribution, and then mask all of these comparisons using a masked comparison circuit based on a carry-save binary adder like Kogge–Stone. This approach was first described by Gérard and Rossi in their masked implementation of qTESLA [GR19], and then reused in subsequent works including [KLS+23, GG25]. This basic framework has recently received a substantial improvement in the work of Eid et al. [EGRZ26], who reduced the number of masked comparisons in these table-based samplers using a binary comparison tree: instead of carrying out masked comparisons with all the elements in the CDT, they use a binary tree with only logarithmically many comparisons, selecting the tree node to compare with in masked form based on the masked results of the earlier comparisons. While this provides a significant asymptotic improvement, especially for larger CDTs, the resulting masked gadget still suffers from significant overhead.

Further related work includes the attempt at a t-probing secure discrete Gaussian sampler with subquadratic complexity in the masking order proposed by the authors of the Mitaka signature scheme [EFG+22, §7], which was later shown by Prest to be flawed [Pre23]. Some heuristic countermeasures to reduce the leakage in unprotected CDT-based samplers have also been suggested in attack papers including [GMRR22, LZY+25], but they do not provide security in the t-probing model. Finally, a series of recent works have attempted to describe a provably secure masked version of the entire Falcon signature scheme, including its discrete Gaussian sampler [CC24, BPT+24, BT25]; however, since these works focus on masking the emulated floating point arithmetic in Falcon, the RCDT base sampler is omitted from the analysis, and its implementation in the companion code to [BPT+24] is not only unmasked, but not even constant-time.

1.2 Our results: introducing Maskaglia

In this paper, we introduce a new discrete Gaussian sampler that combines uniform and geometric distributions with rejection sampling, and gives rise to major efficiency improvements in masked form compared to the state of the art. As an illustration, we show how our techniques can be

used to mask the discrete Gaussian samplers in the HAWK digital signature scheme [BBD+24] (currently a Round 2 candidate additional signature in the NIST standardization process). We obtain masked gadgets for these samplers with provable security and composability in the t-probing model (they satisfy the "probe-isolating non-interference" property of Cassiers–Standaert [CS20]) which outperform previous approaches by a considerable margin.

Concretely, taking the number of used masked AND gates as the traditional cost metric for masked gadgets, our techniques lead to gadgets for the masked samplers in HAWK that are around 20 times better than the previous best work by Eid et al. [EGRZ26], and 40 to 50 times better than the classic CDT-based masked sampler of Gérard and Rossi [GR19]. This is in spite of the fact that Eid et al.'s approach can be shown to achieve the minimal possible number of comparisons among CDT-based techniques.

Furthermore, following the observation of Bronchain and Cassiers [BC22] that the optimal masked adder circuit is *not* in fact a carry-save adder like Kogge–Stone but a plain ripple-carry adder in *bitsliced* form, we note that those previous works, which rely extensively on masked comparisons, can be greatly improved with bitslicing (which almost always benefits discrete Gaussian sampling straightforwardly due to the fact that practical cryptosystems sample discrete Gaussians in batches of several hundreds to thousands). Nevertheless, even if we modify those previous works to use bitslicing, replace their comparison circuits with the optimal ones from [BC22], etc., our approach remains 4–5 times better than the improved version of Eid et al., and 6–8 times better than the improved version of Gérard–Rossi.

1.3 Technical overview

The starting point of our MASKAGLIA approach[3] is the following rejection-based sampler for the (continuous) standard centered normal distribution:

1. Sample two independent exponential random variables $X, Y \sim \mathrm{Exp}(\lambda)$ (with density $\lambda \exp(-\lambda x)$ for $x \geqslant 0$).
2. If $2\lambda X \geqslant (Y-\lambda)^2$, then return $\varepsilon \cdot Y$ for a uniform random sign $\varepsilon \in \{\pm 1\}$, else restart.

We first observe that this sampler can be adapted to discrete Gaussian sampling: if we define $\mathrm{DL}_c(\lambda)$ to be the ("discrete Laplace") distribution over the integers with probability mass function proportional to $\exp(-\lambda|y-c|)$ for $y \in \mathbb{Z}$, the following algorithm samples from the discrete Gaussian distribution $D_{\mathbb{Z},c,\sigma}$ with $\sigma = s/\sqrt{2\lambda}$:

1. Sample two independent random variables $X \sim \mathrm{Exp}(\lambda)$ and $Y \sim \mathrm{DL}_c(\lambda)$.
2. If $X \geqslant \left(|Y-c|/s - s/2\right)^2$, then return Y, else restart.

Now sampling from $\mathrm{DL}_c(\lambda)$ can easily be done with a sample from the geometric distribution $\mathrm{Geom}(\lambda)$ and a Bernoulli trial with known probability, and the former is particularly easy for $\lambda = \ln 2$ (where it can be expressed as the index of the first 1 in the binary expansion of a uniform random bit string).

The exponential distribution is a priori less pleasant, but it can itself be expressed as the sum of a geometric sample (again easy to obtain for $\lambda = \ln 2$) as its integer part X_e, and the base 2 logarithm of a uniform value U in $[1,2)$ as its fractional part X_f.

Then, restricting interest to the case when the parameters s and c are rational numbers (which we can always reduce to by taking sufficiently close approximations), the acceptance condition

[3] As a point of historical trivia, this sampler appears in many references including Devroye [Dev86, Ex. IX.1.4], Abramowitz and Stegun [AS72, 26.8.6.a(4)] and Butcher [But61], with conflicting attributions (sometimes to von Neumann, sometimes to H. Kahn, sometimes to no one in particular), and does *not* seem to originate from G. Marsaglia. However, partly because of its similarity to another sampler of Marsaglia for *tails* of the normal distribution [Mar63], partly out of reverence for Marsaglia's considerable contributions to the problem of random sampling, and partly for the pun, we chose to name our techniques after him.

can be expressed in terms of the integers Y and X_e and the real number X_f, in such a way that the outcome of the comparison only depends on which interval X_f belongs to in a certain finite subdivision of $[0,1)$ (or equivalently, which intervals U belongs to in the corresponding finite subdivision of $[1,2)$). Thus, the sampling of X_f can be effectively replaced by the sampling over a finite set, with probabilities proportional to the lengths of the intervals in the subdivision of $[1,2)$.

This high level description may seem contrived, but does result in a surprisingly efficient discrete Gaussian sampler (especially, but not only, when the parameters s and c have relatively low numerators and denominators). Moreover, this approach is highly masking-friendly, mainly due to the non-obvious but crucial property that, when sampling in masked form a geometric random variable G like the one used to compute Y, essentially any function of G depending on no other secret value can be obtained in masked form at no additional cost (i.e., with no additional masked AND gate) compared to G itself. Indeed, if G can be truncated to values up to n, it is obtained by sampling n masked bits and computing the masked ANDs of all prefixes of the n-bit string and selecting its value accordingly, but the same selection process can be applied to any public function of G in the same way.

Another important observation is that we can generally get away with relatively few bits of randomness to sample the various required distributions, as shown using careful Rényi divergence estimates in the style of [Pre17].

Overall, for the discrete Gaussians used in Hawk (which correspond essentially to $s = 3/2$ and $c \in \{0, 1/2\}$), we obtain masked sampling gadgets that, with bitslicing on an architecture with 32-bit machine words, require less than 5 masked ANDs per sample on average, amortized over the bitsliced parallelism. This is in contrast with around 20 (resp. 32) amortized masked ANDs per sample for our optimized, bitsliced variants of Eid et al.'s gadgets (resp. Gérard and Rossi's), and around 90 (resp. 200) for the actual proposed versions that do not use bitslicing.

1.4 Limitations and future work

Our Maskaglia approach offers an attractive way of masking the discrete Gaussian distributions that appear in a large class of lattice-based cryptosystems. Nevertheless, it does not solve the problem of masked discrete Gaussian sampling in full generality, due to several limitations.

First, the repetition rate of our sampler increases rapidly with the standard deviation of the distribution, so it should not be used as is for wide Gaussians like those of Bliss, Haetae or threshold signatures including [EKT24, BKL+25]. Instead, one can use our sampler to generate relatively narrow base Gaussians and use convolution or rejection techniques to increase their widths. This is already the typical approach used in concrete implementations of such schemes, so the adaptation is fairly straightforward.

Second, this work focuses on the case of discrete Gaussians with fixed, public centers[4] and standard deviations, so that it does not straightforwardly apply to schemes like Falcon and Mitaka that involve sampling Gaussians with secret, variable parameters. Our approach could in principle be adjusted to accommodate variable parameters assuming that those parameters are limited to a relatively short list for which suitable precomputations can be carried out, e.g, as part of key generation (like the standard deviations in Falcon's SamplerZ), but this would not be for free, as most computations involving public constants would need to be replaced by costlier masked

[4] It may seem like Hawk itself also uses discrete Gaussians with secret centers, since the centers (either 0 or 1/2) are obtained by multiplying a known vector (the message hash digest) with the secret basis matrix. This would be true if for each coefficient of the signature, Hawk sampled either the 0- or 1/2-centered discrete Gaussian, but this would already be an issue for the non-masked CDT-based sampler in Hawk's reference implementation, as the center would then be revealed through cache side-channels based on which of the two CDT tables was accessed to generate the sample. Instead, Hawk actually samples from *both* distributions for each coefficient, and selects the sample associated with the appropriate center in constant time. The masked approach is the same: sample using both masked gadgets, and select the correct masked sample using a masked AND gate with the corresponding bit of the secret center vector.

operations. Moreover, in MASKAGLIA, the repetition rate of the sampler depends on the center and standard deviation, and this is a source of leakage if those values are secret: therefore, the approach would need to further incorporate adaptive rejection as in [HPRR20]. A detailed analysis of such a construction is left as future work. This is not to say, however, that MASKAGLIA in its current form is inapplicable to FALCON at all: indeed, it can be used directly to mask FALCON's BaseSampler, and could therefore be plugged in as is within the proposed masked implementation of the emulated FPU-based sampler [BPT+24] (which currently lacks a masked BaseSampler). The other masked operations are likely to be the bottleneck, however.

Third, our approach specifically targets discrete Gaussian sampling, and as such is not directly applicable to schemes that rely on distributions that resemble but do not match actual discrete Gaussians (an example would be rounded Gaussians [HLS18], which occur, e.g., in FRODOKEM [NAB+20]). For certain schemes of that type, it may be fine to use an actual discrete Gaussian as a substitute, since it often leads to parameters that are at least as good and hence compatible, but if the exact "discrete Gaussian-like" distribution is required, our approach would need to be substantially tweaked, perhaps with new rejection identities, or a tight divergence-based analysis showing that a suitably tuned instantiation achieves negligible distance without introducing expensive high-precision operations. This is also left for future work.

Finally, aside from the setting of masked implementations, a natural question that we do not address in this paper is whether our approach provides a competitive constant-time Gaussian sampler *without masking*, when compared to a constant-time CDT-based sampler, for example. We expect the answer to be platform-dependent. On desktop and server CPUs, it seems that the cost of constant-time discrete Gaussian sampling is typically dominated by the generation of the random bits involved; since our sampler uses somewhat more randomness than plain CDT-based sampling on average (around 50% more for the Gaussians in HAWK), we are unlikely to outperform in that setting. However, we need fewer memory accesses, so we may achieve superior performance on smaller platforms with costlier memory loads (e.g., cache-less embedded platforms). A detailed analysis is again left as future work.

2 Notations and preliminaries

2.1 Distributions, and basic sampling facts

Throughout the paper log denotes the binary logarithm, while ln denotes the natural logarithm. For $x \in \mathbb{R}$, we denote by $\lfloor x \rfloor$ the floor, and by $\{x\} = x - \lfloor x \rfloor \in [0,1)$ the fractional part.

Sets, measures, and basic notation. For $a \le b$, we write $\mathrm{Unif}([a,b])$ for the continuous uniform distribution on $[a,b]$. We use the shorthand $X \xleftarrow{\$} \mathcal{D}$ to mean that the random variable X is sampled from the distribution $\mathcal{D}$. Throughout the paper we consider two supports: $\mathcal{S} \in \{\mathbb{R}, \mathbb{Z}\}$. When $\mathcal{S} = \mathbb{R}$ we use the Lebesgue measure, and when $\mathcal{S} = \mathbb{Z}$ we use the counting measure. To unify notation, we write $d\mu$ for the relevant base measure on $\mathcal{S}$. We write $A \propto B$ to indicate equality up to a multiplicative constant that does not depend on the relevant variable.

Kernels and normalization. Distributions in this paper are most conveniently specified via an unnormalized nonnegative function on the support.

Definition 1 (Kernel-defined distribution). *Let $\mathcal{S} \in \{\mathbb{R}, \mathbb{Z}\}$ and let $\pi : \mathcal{S} \to \mathbb{R}_{\ge 0}$ be measurable with finite, positive normalization constant*

$$C_\pi := \int_{\mathcal{S}} \pi(t)\, d\mu(t) \in (0, \infty).$$

We write $\mathcal{D}_\pi$ for the probability distribution on $\mathcal{S}$ with density/mass function

$$f_\pi(x) := \frac{\pi(x)}{C_\pi} \qquad (x \in \mathcal{S}),$$

Table 1. Kernel presentations of distributions used in this work. For each row, the density/mass function with respect to $d\mu$ is $f(x) = \pi(x)/C_\pi$.

Notation	Name	$\mathcal{S}$	Kernel $\pi(x)$	Normalizer C_π
$\mathrm{Bern}(p)$	Bernoulli	$\{0,1\}$	$p^x(1-p)^{1-x}\mathbf{1}_{x\in\{0,1\}}$	1
$\mathrm{Unif}([a,b])$	Uniform	$\mathbb{R}$	$\mathbf{1}_{a\le x\le b}$	$b-a$
$\mathcal{N}(c,\sigma^2)$	Continuous Gaussian	$\mathbb{R}$	$\exp\left(-\frac{(x-c)^2}{2\sigma^2}\right)$	$\sqrt{2\pi}\,\sigma$
$D_{\mathbb{Z},c,\sigma}$	Discrete Gaussian	$\mathbb{Z}$	$\exp\left(-\frac{(x-c)^2}{2\sigma^2}\right)$	$\rho_\sigma(\mathbb{Z}-c) := \sum_{t\in\mathbb{Z}}\exp\left(-\frac{(t-c)^2}{2\sigma^2}\right)$
$\mathrm{Lap}_c(\lambda)$	Continuous Laplace	$\mathbb{R}$	$\exp(-\lambda\lvert x-c\rvert)$	$2/\lambda$
$\mathrm{DL}_c(\lambda)$	Discrete Laplace	$\mathbb{Z}$	$\exp(-\lambda\lvert x-c\rvert)$	$\mathcal{C}_{c,\lambda} := \sum_{t\in\mathbb{Z}}\exp(-\lambda\lvert t-c\rvert)$
$\mathrm{Exp}(\lambda)$	Exponential	$\mathbb{R}$	$e^{-\lambda x}\mathbf{1}_{x\ge 0}$	$1/\lambda$
$\mathrm{Geom}(p)$	Geometric	$\mathbb{Z}^+$	$(1-p)^x\,\mathbf{1}_{x\ge 0}$	$1/p$

with respect to $d\mu$.

In Table 1, we collect the distributions appearing in this paper with their relevant kernel and normalizer.

Inverse transform sampling. We recall the standard inverse transform principle for completeness. Let F be a cumulative distribution function (CDF) on $\mathbb{R}$. Its generalized inverse (right-continuous quantile function) is

$$F^{-1}(u) := \inf\{x \in \mathbb{R} \mid F(x) \ge u\}, \qquad 0 < u < 1.$$

Lemma 1 (Inverse transform sampling). *If $U \sim \mathrm{Unif}([0,1])$, then $F^{-1}(U)$ has CDF F.*

Indeed, for any $x \in \mathbb{R}$, by definition of the generalized inverse and right-continuity of F,

$$\{u \in (0,1) : F^{-1}(u) \le x\} = \{u \in (0,1) : u \le F(x)\}.$$

Therefore,

$$\Pr[F^{-1}(U) \le x] = \Pr[U \le F(x)] = F(x),$$

since $\Pr[U \le u] = u$ for $U \sim \mathrm{Unif}([0,1])$.

Rényi divergence. For two distributions $\mathcal{P},\mathbb{Q}$ such that $\mathsf{Supp}(\mathcal{P}) \subset \mathsf{Supp}(\mathbb{Q})$ and $a \in (1,+\infty)$ we define the Rényi divergence of order a as:

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

To simplify notations, for any random variables X,Y such that $\mathsf{Supp}(X) \subset \mathsf{Supp}(Y)$ we also write $R_a(X\|Y)$ to denote the Rényi divergence of order a of their underlying distributions.

2.2 Masking backgroud

Masking and t-probing model. Masking is a widely adopted countermeasure designed to protect sensitive data against side-channel analysis. It operates by partitioning a secret variable x into d separate components x_i, referred to as *shares*, such that $\bigoplus_{i=0}^{d-1} x_i = x$. Consequently, any subset of $d-1$ shares remains independent of the secret x. Maintaining this property throughout the execution of a circuit for all sensitive intermediate values is necessary for security. This requirement is formalized by the t-probing model [ISW03], which provides a security guarantee that an adversary observing up to $t = d-1$ intermediate variables learns no information about the secret. In this work, the variable d always denotes the number of shares.

In a masked implementation, standard functions are replaced by masked *gadgets* that process shared inputs and produce shared outputs. Linear operations, such as `NOT` and `XOR`, can be performed share-wise, resulting in an overhead that is linear in the number of shares d. Conversely, non-linear operations, such as an `AND` gate between two shared values, incur a quadratic overhead and require the generation of fresh randomness. In practice, the computational cost of these non-linear operations dominates the total execution time, even at relatively low masking orders.

In this work, we establish the probing security of our constructions using the Probe-Isolating Non-Interference (PINI) property [CS20]. Beyond ensuring the probing security of individual gadgets, PINI guarantees trivial composability: PINI-compliant gadgets can be chained without additional randomness refreshes while preserving the security of the resulting circuit. Notably, linear operations are natively PINI.

Bitsliced implementation. To achieve high performance in software, a common masked implementation technique is *bitslicing* [Bih97, GLSV15]. Bitslicing consists of reorienting the data representation such that each w-bit register holds the i-th bit of w different sensitive variables. This approach is particularly advantageous for Boolean masking, as standard bitwise instructions (e.g., `AND`, `XOR`) effectively act as w parallel 1-bit logic gates. Consequently, while our gadgets are described with single-bit gates, they are designed to fully exploit the processor's word size, allowing for the simultaneous processing of multiple samples and significantly reducing the amortized cost of non-linear operations.

3 New algorithms for discrete Gaussian sampling

We present a single rejection-based template that turns a *Laplace kernel* into a *Gaussian kernel* on the same support, either $\mathbb{R}$ (continuous) or $\mathbb{Z}$ (discrete). The proposal sample is Laplace (i.e., it follows a symmetrized exponential or geometric distribution) and the acceptance step applies a quadratic correction in the exponent that cancels the linear term exactly. As discussed in the introduction, this comes from a classical sampler for the normal distribution that dates back at least to the 1960s, although the extension to discrete Gaussians is probably a new observation.

The key observation. Let $r \geq 0$, for $\lambda > 0$ and $s > 0$, then

$$\exp\left(-\frac{\lambda}{s^2}r^2\right) = \exp\left(-\lambda r\right) \cdot \exp\left(-\lambda\left(\frac{r}{s} - \frac{s}{2}\right)^2\right) \Big/ \exp\left(-\frac{\lambda s^2}{4}\right).$$

This identity entails that multiplying a Laplace kernel $e^{-\lambda r}$ by an acceptance factor $e^{-\lambda\left(\frac{r}{s}-\frac{s}{2}\right)^2}$ yields a Gaussian kernel $e^{-(\lambda/s^2)r^2}$ up to a constant:

$$\underbrace{\exp\left(-\frac{\lambda}{s^2}r^2\right)}_{\text{Gaussian kernel, width } s/\sqrt{2\lambda}} \propto \overbrace{\exp(-\lambda r)}^{\text{Laplace kernel, rate } \lambda} \cdot \underbrace{\exp\left(-\lambda\left(\frac{r}{s} - \frac{s}{2}\right)^2\right)}_{\text{rejection rate } \alpha(r)}$$

Implementing a naive rejection sampler via a single auxiliary uniform $U \sim \mathrm{Unif}(0,1)$ gives the event $U \le \alpha(Y)$ for Y the Laplace sample. Equivalently $-\ln U \ge \lambda(\frac{Y}{s} - \frac{s}{2})^2$. Set $E = -\frac{\ln U}{\lambda} \sim \mathrm{Exp}(\lambda)$, so that we can write the rejection event simply as

$$E \ \ge \ \left(\frac{Y}{s} - \frac{s}{2}\right)^2.$$

Hence, we can correct a Laplace kernel into a Gaussian one with a simple quadratic test with an exponential proxy.

3.1 A unified $\{\mathbb{R}, \mathbb{Z}\}$-Gaussian sampler from Laplace

We begin with a correctness lemma that covers both the continuous and discrete settings.

Lemma 2 (Laplace-to-Gaussian transform). *Fix $\lambda > 0$, $s > 0$, and $c \in \mathbb{R}$. Let the support be either $\mathcal{S} = \mathbb{R}$ (with Lebesgue measure) or $\mathcal{S} = \mathbb{Z}$ (with counting measure). Let Y be a Laplace random variable on $\mathcal{S}$ of center c and rate λ, i.e., with unnormalized density/mass: $\pi_{\mathrm{Lap}_c(\lambda)}(y) \ \propto \ \exp(-\lambda|y-c|), y \in \mathcal{S}$. Independently sample $X \sim \mathrm{Exp}(\lambda)$ and accept Y iff*

$$X \ \ge \ \left(\frac{|Y-c|}{s} - \frac{s}{2}\right)^2.$$

Let Z denote the accepted output. Then:

1. *The law of Z is Gaussian on the same support, i.e., with unnormalized kernel $\pi_{\mathrm{G}}(y) \ \propto \ \exp\left(-\frac{\lambda}{s^2}|y-c|^2\right)$.*
2. *The acceptance probability is*

$$p_{\mathrm{acc}} = e^{-\lambda s^2/4} \cdot \frac{C_{\mathrm{G}}}{C_{\mathrm{Lap}}},$$

where $C_{\mathrm{Lap}} = \int_{\mathcal{S}} e^{-\lambda|t-c|}\, d\mu(t)$, and $C_{\mathrm{G}} = \int_{\mathcal{S}} e^{-(\lambda/s^2)|t-c|^2}\, d\mu(t)$, and $d\mu$ is the base measure on $\mathcal{S}$.

Proof. Fix $y \in \mathcal{S}$. Observe that the quantity $\left(\frac{|y-c|}{s} - \frac{s}{2}\right)^2$ is nonnegative, hence admissible as a rejection threshold for an $\mathrm{Exp}(\lambda)$ variable. Conditioned on $Y = y$, the algorithm accepts with probability

$$\Pr[\text{accept} \mid Y = y] = \Pr\left[X \ge \left(\frac{|y-c|}{s} - \frac{s}{2}\right)^2\right] = \exp\left(-\lambda\left(\frac{|y-c|}{s} - \frac{s}{2}\right)^2\right).$$

Therefore the (unnormalized) output density/mass is

$$\pi_{\mathrm{Lap}}(y) \cdot \Pr[\text{accept} \mid Y = y] \ \propto \ e^{-\lambda|y-c|} \cdot \exp\left(-\lambda\left(\frac{|y-c|}{s} - \frac{s}{2}\right)^2\right).$$

This proves (1). Integrating/summing the accepted unnormalized mass yields (2). □

Lemma 2 yields two samplers that are literally the same program, differing only in whether the Laplace proposal is continuous or discrete. We compile them side by side in Figure 1.

Corollary 1. *Fix $\lambda > 0$, $s > 0$, and $c \in \mathbb{R}$, and set $\sigma = s/\sqrt{2\lambda}$.*

1. *Algorithm 1 outputs $\mathcal{N}(c, \sigma^2)$ with acceptance probability*

$$p_{\mathrm{acc}}^{\mathbb{R}} = \frac{s\sqrt{\pi\lambda}}{2}\, e^{-\frac{\lambda s^2}{4}}.$$

Alg. 1 :: Continuous Gaussian template

Input: $c \in \mathbb{R}$, $s > 0$, $\lambda > 0$
Output: $Z \sim \mathcal{N}(c, \sigma^2)$

repeat
  $X \xleftarrow{\$} \mathrm{Exp}(\lambda)$
  $Y \xleftarrow{\$} \mathrm{Lap}_c(\lambda)$
until $X \geq (|Y-c|/s - s/2)^2$
return Y

Alg. 2 :: Discrete Gaussian template

Input: $c \in \mathbb{R}$, $s > 0$, $\lambda > 0$
Output: $Z \sim D_{\mathbb{Z},c,\sigma}$

repeat
  $X \xleftarrow{\$} \mathrm{Exp}(\lambda)$
  $Y \xleftarrow{\$} \mathrm{DL}_c(\lambda)$
until $X \geq (|Y-c|/s - s/2)^2$
return Y

Fig. 1. Rejection based sampling on $\mathbb{R}$ and on $\mathbb{Z}$ for Gaussian from Laplace with $\sigma = s/\sqrt{2\lambda}$

2. *Algorithm 2 outputs $D_{\mathbb{Z},c,\sigma}$ with acceptance probability*

$$p_{\mathrm{acc}}^{\mathbb{Z}} = e^{-\lambda s^2/4} \cdot \frac{\sum_{z\in\mathbb{Z}} \exp\left(-\frac{\lambda}{s^2}(z-c)^2\right)}{\sum_{z\in\mathbb{Z}} \exp(-\lambda|z-c|)} \geqslant e^{-\frac{\lambda}{4s^2}} \tanh\left(\frac{\lambda}{2}\right) e^{-\frac{\lambda s^2}{4}}$$

In both cases, the number of iterations is geometric with success probability p_{acc}. Hence the expected number of iterations needed to output a sample is $\mathbb{E}[\#iters] = 1/p_{\mathrm{acc}}$.

The proof of Corollary 1 is a straightforward computation and can be found in Section A.

Remark 1 (on efficiency). The target Gaussian depends only on the ratio λ/s^2 (equivalently, on σ). For a fixed target σ, one may choose λ (e.g. $\lambda = \ln 2$ for fast discrete primitives) and then set $s = \sigma\sqrt{2\lambda}$. The acceptance probability depends only on that choice. Furthermore, one can note that for $\mathcal{S} = \mathbb{R}$, the function $p_{\mathrm{acc}}^{\mathbb{R}}(s)$ is maximized at $s = \sqrt{2/\lambda}$, yielding $p_{\mathrm{acc}}^{\max} = \sqrt{\pi/2}\, e^{-1/2} \approx 0.759$.

3.2 Exact instantiations: Laplace and exponential base samplers

The two samplers differ only in how the Laplace proposal is sampled; both also require an exact sampler for $\mathrm{Exp}(\lambda)$. Let us now fully explicit samplers for these distributions.

Continuous Laplace, from an exponential and a sign. The continuous Laplace is easily obtained from an exponential sample and a uniform sign flip.

Lemma 3. *If $E \sim \mathrm{Exp}(\lambda)$ and $B \in \{\pm 1\}$ is uniform and independent, then $c + BE \sim \mathrm{Lap}_c(\lambda)$.*

Proof. Clear from the pdf of the Laplace distribution and of the exponential. □

Thus each iteration of Algorithm 1 uses two exponential draws (one for X, one for E) and one random sign.

Discrete Laplace sampler (specializing to $\lambda = \ln 2$). Let $\lambda > 0$ and $c \in \mathbb{R}$. Recall that the discrete Laplace distribution $\mathrm{DL}_c(\lambda)$ on $\mathbb{Z}$ is defined by

$$\Pr[Y = y] = \frac{e^{-\lambda|y-c|}}{\mathscr{C}_{c,\lambda}}, \qquad \mathscr{C}_{c,\lambda} = \sum_{z\in\mathbb{Z}} e^{-\lambda|z-c|}.$$

Only the fractional part of c matters up to an integer shift. Writing $c = n + \delta$ with $n = \lfloor c \rfloor \in \mathbb{Z}$ and $\delta \in [0,1)$,

$$Y \sim \mathrm{DL}_c(\lambda) \quad \Longleftrightarrow \quad Y - n \sim \mathrm{DL}_\delta(\lambda).$$

Alg. 3 :: Discrete Laplace sampler

Input: $c \in \mathbb{R}$
Output: $Y \sim \mathrm{DL}_c(\ln 2)$

$n \leftarrow \lfloor c \rceil$; $\delta \leftarrow c - n$ // $\delta \in [0,1)$
$G \xleftarrow{\$} \mathrm{Geom}(1/2)$ // $\Pr[G = k] = 2^{-(k+1)}$
$S \xleftarrow{\$} \mathrm{Bern}\left(\frac{2^{\delta}}{2^{\delta} + 2^{1-\delta}}\right)$
if $S = 1$ **then return** $n + (G + 1)$
else return $n - G$

Alg. 4 :: Exponential sampler for $\lambda = \ln 2$

Output: $X \sim \mathrm{Exp}(\ln 2)$

$K \xleftarrow{\$} \mathrm{Geom}(1/2)$
$V \xleftarrow{\$} \mathrm{Unif}([1/2,1])$
$U \leftarrow -\frac{\ln V}{\ln 2}$ // $U \sim \mathrm{Exp}(\ln 2)$ on $[0,1)$
return $K + U$

For $\delta \in [0,1)$,

$$\sum_{z \in \mathbb{Z}} e^{-\lambda |z-\delta|} = \frac{e^{-\lambda\delta} + e^{-\lambda(1-\delta)}}{1 - e^{-\lambda}}.$$

For $\lambda = \ln 2$ this becomes $2^{1-\delta} + 2^{\delta}$, and the unnormalized weights are powers of two: $\Pr[Y = y] \propto 2^{-|y-\delta|}$. Splitting at the boundary between $\{0\}$ and $\{1\}$ yields side probabilities

$$\Pr[Y \geq 1] = \frac{2^{\delta}}{2^{\delta} + 2^{1-\delta}}, \qquad \Pr[Y \leq 0] = \frac{2^{1-\delta}}{2^{\delta} + 2^{1-\delta}},$$

and, conditional on the chosen side, the distance from the boundary is geometric (as the conditional mass function of a geometric remains proportional to $e^{-\lambda x}$). All in all, this gives a simple algorithm using only Geometric and Bernoulli samplers, written in Algorithm 3.

Sampling from $\mathrm{Exp}(\lambda)$ *via integer and fractional parts.* Let $X \sim \mathrm{Exp}(\lambda)$. Write

$$X = K + U, \qquad K = \lfloor X \rfloor \in \mathbb{Z}_{\geq 0}, \quad U = \{X\} \in [0,1).$$

Then K and U are independent with

$$\Pr[K = k] = (1 - e^{-\lambda})e^{-\lambda k}, \qquad f_U(u) = \frac{\lambda e^{-\lambda u}}{1 - e^{-\lambda}}, \quad u \in [0,1).$$

Thus $K \sim \mathrm{Geom}(p)$ with $p = 1 - e^{-\lambda}$, and U is an exponential truncated to $[0,1)$. In the special case $\lambda = \ln 2$, we have $p = 1/2$. We can sample the truncated variable U using the inverse transform method (see Section 2.1) simply by restricting the uniform input range: if $V \sim \mathrm{Unif}([1/2,1])$, then $U = -\frac{\ln V}{\ln 2}$ follows the required distribution on $[0,1)$. This gives rise to Algorithm 4.

Wrapping up. For completeness, we spell out Algorithm 2 when $\lambda = \ln 2$ and the proposal is sampled by Algorithm 3. The output is a discrete Gaussian with variance $\sigma^2 = s^2/(2\ln 2)$.

Corollary 2. *For $s > 0$, and $c \in \mathbb{R}$, and set $\sigma = \frac{s}{\sqrt{2\ln 2}}$. Algorithm 2 outputs $D_{\mathbb{Z},c,\sigma}$ with an expected number of iterations:*

$$\mathbb{E}[\#\mathit{iters}] \leqslant 2^{s^2/4} \max\left\{3,\ 2^{3/2+1/(4s^2)}\right\}.$$

Proof. Correctness follows from the correctness of the template algorithm and of the correctness of the Geometric and discrete Laplace samplers. Writing $r = \{c\}$, we have that $\sum_{z\in\mathbb{Z}} 2^{-|z-c|} = 2^r + 2^{1-r}$ for the denominator and $\sum_{z\in\mathbb{Z}} 2^{-(z-c)^2/s^2} \geqslant 2^{-\delta^2/s^2}$ for the numerator. Hence

$$p_{\mathrm{acc}}^{\mathbb{Z}} \geqslant 2^{-s^2/4} \cdot \frac{2^{-\delta^2/s^2}}{2^r + 2^{1-r}}.$$

Now minimize over $r \in [0,1)$. Using symmetry, we can minimize over $r \in [0, \frac{1}{2}]$. Because the RHS has a concave log on $[0, \frac{1}{2}]$, the minimum occurs at an endpoint $r \in \{0, \frac{1}{2}\}$, giving the result. □

Remark 2. The bound grows quickly with s. For large standard deviations, one can instead use the convolution trick (e.g. [Pei10, Thm. 3.1]) to fall back on sampling smaller Gaussians: if $s/\sqrt{N}$ exceeds the smoothing parameter, then summing N independent discrete Gaussians with parameter $s/\sqrt{N}$ yields a distribution statistically close to a discrete Gaussian of parameter s.

Where transcendental functions enter. The rejection decision compares an exponential variate to a threshold; it does not compute e^{-t}. However, Algorithm 4 uses $\ln(\cdot)$ to sample the fractional part U of $\mathrm{Exp}(\ln 2)$. We next show how, when s and c are rational, one can remove transcendentals *at runtime from the acceptance test* by reducing it to *only one* exact threshold comparison.

4 Practical Discrete Gaussian samplers

We now focus on practical improvements for this sampler: replace the continuous exponential sampling by a discrete proxy that is integer-friendly and tailcutting all the geometric distributions appearing in the computation. Indeed, in practice even though the discrete Gaussian distribution has unbounded support, we can only represent it up to a tailcut and forget about the negligible tails. The same goes for the Geometric distribution appearing at runtime, so we need to precisely handle the bias it produces.

4.1 Log-free acceptance via finite threshold comparisons

Throughout this subsection we fix $\lambda = \ln 2$ (so that the integer part of $\mathrm{Exp}(\lambda)$ is $\mathrm{Geom}(1/2)$). Given rational s and c, we want to replace the real-valued fractional draw U which is exclusively used in the acceptance test $K + U \geq (|Y - c|/s - s/2)^2$ by a discrete proxy used only through comparisons, preserving acceptance probabilities exactly.

Write

$$s = \frac{p}{q} \in \mathbb{Q}_{>0}, \qquad c = \frac{p'}{q'} \in \mathbb{Q}, \qquad (p, q, p', q' \in \mathbb{Z},\ q, q' > 0).$$

Expression of the rejection test in terms of integers. Fix $Y = y \in \mathbb{Z}$. Using $|y - c| = \frac{|yq' - p'|}{q'}$ and $s = p/q$,

$$\frac{|y-c|}{s} - \frac{s}{2} = \frac{|yq' - p'|}{q'} \cdot \frac{q}{p} - \frac{p}{2q} = \frac{2q^2|yq' - p'| - p^2q'}{2pqq'}.$$

Define $\tilde{L}_y := 2q^2\,|yq' - p'| - p^2q'$ and $\tilde{K}_0 := (2pqq')^2$. Let $g := \gcd(2q^2,\ p^2q') \in \mathbb{Z}_{\geq 1}$, then $g \mid \tilde{L}_y$ for every $y \in \mathbb{Z}$, hence $g^2 \mid \tilde{L}_y^2$. Set the reduced $L_y := \tilde{L}_y/g$, and $K_0 := \tilde{K}_0/g^2$. Then the rejection condition is equivalent to

$$K_0(K + U) < L_y^2.$$

The only remaining *real* quantity is $U \in [0,1)$, and it only appears through $K_0U \in [0, K_0)$.

Replacing U by an integer proxy. Define the integer-valued random variable

$$W := \lfloor K_0U \rfloor \in \{0,1,\ldots,K_0 - 1\}. \tag{1}$$

Since U has a continuous density on $[0,1)$, $\Pr[K_0U \in \mathbb{Z}] = 0$. Hence for every integer $r \in \{0,1,\ldots,K_0\}$,

$$\Pr[K_0U < r] = \Pr[\lfloor K_0U \rfloor < r] = \Pr[W < r].$$

Therefore the rejection is equivalent almost surely to

$$K_0K + W < L_y^2. \tag{2}$$

Now write the Euclidean decomposition

$$L_y^2 = K_0A_y + R_y, \qquad A_y = \left\lfloor \frac{L_y^2}{K_0} \right\rfloor \in \mathbb{Z}_{\geq 0}, \qquad R_y = L_y^2 \bmod K_0 \in \{0,\ldots,K_0 - 1\}.$$

Fix y and $k = K$. Since $W \in \{0,\ldots,K_0 - 1\}$,

$$K_0k + W < K_0A_y + R_y \iff \begin{cases} \text{always true} & \text{if } k < A_y, \\ \text{always false} & \text{if } k > A_y, \\ W < R_y & \text{if } k = A_y. \end{cases}$$

Thus, for each y, the dependence on the fractional part concentrates into a single boundary case $k = A_y$, where the test requires only $W < R_y$. Define the set of residues that can occur:

$$\mathcal{R}_{s,c} := \{R_y :\ y \in \mathbb{Z}\} \subseteq \{0,1,\ldots,K_0 - 1\}.$$

The acceptance test only queries W via comparisons of the form $W < r$ for $r \in \mathcal{R}_{s,c}$. For convenience we also include endpoints: $\mathcal{T}_{s,c} := \mathcal{R}_{s,c} \cup \{0, K_0\}$ if they are not already in $\mathcal{R}_{s,c}$.

Interlude: number of sentinels in $\mathcal{T}_{s,c}$. We now evaluate the number of sentinels in the threshold set. For the proofs of the following lemmas, we refer to Section B, as they are quite lengthy. They rely on counting points in arithmetic progressions and on the enumeration of modular squares.

Proposition 1 (Generic bound). *With the notation of the current section, and setting $M := \frac{2p^2q'}{g} \in \mathbb{Z}_{>0}$, one has*

$$|\mathcal{T}_{s,c}| \ \leq\ \begin{cases} \min\!\left(2M \cdot 2^{-|P|},\ M\right) + 2, & \text{if } g \mid 4p^2, \\ 2M \cdot 2^{-|P|} + 2, & \text{otherwise,} \end{cases}$$

for $P := \{\ell \text{ odd prime} :\ \ell \mid p,\ \ell \nmid q'\}$.

Interestingly, the driving force in these bounds is *only* the numerator of the width s and the denominator of the center. While these bounds can be apparently quite large, it is never a problem for time complexity; indeed, we show in Section 5.3, that we can do the on-the-fly computations from the discrete Laplace sampler, with no overheads. Memory-wise, even when the number of sentinels is large, only a small number of them are actually reached in practice and implementers can choose whether they prefer to store these in a table or recompute them on-the-fly.

A discrete proxy matching the CDF on $\mathcal{T}_{s,c}$. We now build a discrete random variable D supported on finitely many thresholds whose CDF matches that of W at every queried threshold. This suffices for exact preservation of acceptance probabilities. First compute the CDF of W. For $\lambda = \ln 2$, the fractional part U is $\mathrm{Exp}(\ln 2)$ truncated to $[0,1)$, hence for $t \in [0,1)$,

$$\Pr[U < t] = \frac{1 - 2^{-t}}{1 - 2^{-1}} = 2(1 - 2^{-t}).$$

Therefore, for each $r \in \{0,1,\ldots,K_0\}$,

$$\Pr[W < r] = \Pr\left[U < \frac{r}{K_0}\right] = 2\left(1 - 2^{-r/K_0}\right) = 2 - 2^{1-r/K_0}.$$

Let $0 = r_0 < r_1 < \cdots < r_m = K_0$ be the distinct elements of $\mathcal{T}_{s,c}$ in increasing order.

Definition 2 (Proxy distribution $\mathrm{RED}_{s,c}$). *Define the Rejection-Equivalent Distribution $\mathrm{RED}_{s,c}$ as the distribution supported on $\{r_0, r_1, \ldots, r_{m-1}\}$ with*

$$Pr[\mathrm{RED}_{s,c} = r_j] := 2^{1-r_j/K_0} - 2^{1-r_{j+1}/K_0}, \qquad j = 0,1,\ldots,m-1.$$

Lemma 4 (CDF matching on queried thresholds). *For every $j \in \{0,1,\ldots,m\}$,*

$$Pr[\mathrm{RED}_{s,c} < r_j] = Pr[W < r_j].$$

In particular, $Pr[\mathrm{RED}_{s,c} < r] = Pr[W < r]$ for all $r \in \mathcal{T}_{s,c}$.

Proof. By definition and a telescoping sum,

$$\begin{aligned}\Pr[\mathrm{RED}_{s,c} < r_j] &= \sum_{t=0}^{j-1} \Pr[\mathrm{RED}_{s,c} = r_t] \\ &= \sum_{t=0}^{j-1} \left(2^{1-r_t/K_0} - 2^{1-r_{t+1}/K_0}\right) \\ &= 2^{1-r_0/K_0} - 2^{1-r_j/K_0} = 2 - 2^{1-r_j/K_0}.\end{aligned}$$

This equals to $\Pr[W < r_j]$ by definition. □

Plugging in the proxy: the MASKAGLIA sampler. We can now replace the boundary comparison involving W by one involving the proxy random variable $D \sim \mathrm{RED}_{s,c}$, without changing any conditional acceptance probabilities, as shown in Algorithm 5.

Theorem 1 (Correctness of the proxy acceptance test). *Fix $s = p/q \in \mathbb{Q}_{>0}$ and $c = p'/q' \in \mathbb{Q}$, and define L_y, K_0 as in before. Consider, for a fixed proposal value $Y = y$, the original boundary-randomized test*

$$\text{reject if } K_0 K + W < L_y^2, \quad W = \lfloor K_0 U \rfloor,$$

where $K \sim \mathrm{Geom}(1/2)$ and U is the fractional part of the exponential law $\mathrm{Exp}(\ln 2)$, and the modified test

$$\text{reject if } K_0 K + D < L_y^2, \quad D \sim \mathrm{RED}_{s,c}.$$

Then for every $y \in \mathbb{Z}$, the conditional rejection probability given $Y = y$ is identical in both procedures. Consequently, replacing the original fractional-part comparison by the proxy test in the outer sampler preserves the distribution of the accepted output exactly.

Alg. 5 :: Integer Gaussian sampler

Input: $s = p/q \in \mathbb{Q}_{>0}$, $c = p'/q' \in \mathbb{Q}$
Output: $Z \sim D_{\mathbb{Z},c,\sigma}$, $\sigma = s/\sqrt{2\ln 2}$

1 $\tilde{K}_0 \leftarrow (2pqq')^2$
2 $g \leftarrow \gcd(2q^2, p^2q')$
3 $K_0 \leftarrow \tilde{K}_0/g^2$
4 **repeat**
5 $\quad Y \leftarrow \mathrm{DL}_c(\ln 2)$ // Algorithm 3
6 $\quad L_Y \leftarrow (2q^2\,|Yq' - p'| - p^2q')/g$
7 $\quad K \xleftarrow{\$} \mathrm{Geom}(1/2)$
8 $\quad D \xleftarrow{\$} \mathrm{RED}_{s,c}$
9 **until** $K_0K + D \geq L_Y^2$
10 **return** Y

Proof. Fix $y \in \mathbb{Z}$ and write $L_y^2 = K_0A_y + R_y$ by Euclidean division as before . Fix $k \in \mathbb{Z}_{\geq 0}$. Since $W, D \in \{0, \ldots, K_0 - 1\}$, both comparisons

$$K_0k + W < K_0A_y + R_y \quad \text{and} \quad K_0k + D < K_0A_y + R_y$$

are deterministically true when $k < A_y$ and deterministically false when $k > A_y$. In the boundary case $k = A_y$, the original test rejects with probability $\Pr[W < R_y]$ and the modified test rejects with probability $\Pr[D < R_y]$.

By definition, $R_y \in \mathcal{R}_{s,c} \subseteq \mathcal{T}_{s,c}$, and Lemma 4 implies $\Pr[D < R_y] = \Pr[W < R_y]$. Therefore the conditional rejection probability given $(Y = y, K = k)$ is identical in the two procedures for all (y, k). Averaging over K yields equality conditioned on $Y = y$, hence the accepted-output distribution is unchanged. □

4.2 Final cut: instantiating the sampler

Since we target applications in signature schemes, we analyze the sampler in the context of search games using Rényi divergence. This approach extends naturally to distinguishing games via statistical distance.

Let $\mathcal{D}$ denote the ideal MASKAGLIA distribution (Algorithm 5) and $\tilde{\mathcal{D}}$ denote the instantiated distribution with finite precision and support. Following the argument of [Pre17, Section 3.3], to ensure the concrete instantiation $\tilde{\mathcal{D}}$ incurs a security loss of at most 1 bit over q_s signature queries, it suffices to bound the Rényi divergence of order a (typically $a \in \{256, 512, \infty\}$) as:

$$R_a(\tilde{\mathcal{D}} \| \mathcal{D}) \leqslant 1 + \frac{1}{4q_s} =: t_R.$$

Bounded Domain for Y. The output Y of the ideal sampler has infinite support ($\mathbb{Z}$). For implementation, we enforce a tailcut bound B_g. We restrict the proposal distribution (Line 5) to the interval $(-B_g, B_g)$[5]. Let $\mathcal{D}_{|B_g}$ be the distribution conditional on $|Y| < B_g$. The bound B_g is chosen such that:

$$R_a(\mathcal{D}_{|B_g} \| \mathcal{D}) \leqslant t_R.$$

[5] Since reachable sentinels depend on the values of Y, this cut ensures efficiency even when $\mathcal{T}_{s,c}$ is large.

Implementation: This restriction propagates to the Discrete Laplace sampler (Algorithm 3). Specifically, the geometric sample $G \leftarrow \mathrm{Geom}(1/2)$ used to construct Y must be sampled conditionally on the resulting Y falling within $(-B_g, B_g)$. This effectively replaces the infinite-precision geometric sampler with a finite one, consuming at most B_g bits of fresh entropy.

Bounded Sampling for K. While the variable $K \sim \mathrm{Geom}(1/2)$ in the acceptance step (Line 7) theoretically requires infinite support, we observe that for sufficiently large K, the acceptance condition becomes satisfied deterministically.

Recall the acceptance inequality: $K_0 K + D \geqslant L_Y^2$. Let $L_{\max} = \max\{L_y : |y| < B_g\}$ be the maximum threshold reachable within the tailcut bounds. We define a saturation threshold for K:

$$K_{\mathrm{sat}} := \left\lfloor \frac{L_{\max}^2}{K_0} \right\rfloor + 1.$$

If $K \geqslant K_{\mathrm{sat}}$, then $K_0 K \geqslant L_{\max}^2 \geqslant L_Y^2$. Since $D \geqslant 0$, the inequality $K_0 K + D \geqslant L_Y^2$ holds strictly for all valid Y. Consequently, the sampler can clamp the geometric sampling of K at K_{sat}. Any sample value exceeding K_{sat} is treated as K_{sat}, resulting in immediate acceptance. This optimization is exact and introduces no divergence.

Once this exact bound is found, a more aggressive tailcut can be performed on K by using the same approach as for Y. It is possible to numerically compute the Rényi divergence of the overall distribution with respect to the tailcut on K. The upper bound we computed then serves as a starting point in an exhaustive search for a better bound. We apply this in our parameter script.

Bernoulli instantiation of the comparison. The variable $D \sim \mathrm{RED}_{s,c}$ is only required to resolve the acceptance inequality when K falls strictly on the boundary. For a fixed proposal Y, let us perform the Euclidean division of the threshold L_Y^2 by the modulus K_0:

$$L_Y^2 = q_Y \cdot K_0 + r_Y, \quad \text{where } 0 \leqslant r_Y < K_0. \tag{3}$$

Substituting this into the acceptance check $K_0 K + D \geqslant L_Y^2$:

- If $K > q_Y$, the condition holds (Accept).
- If $K < q_Y$, the condition fails (Reject), as $D < K_0$.
- If $K = q_Y$, the condition simplifies to $D \geqslant r_Y$.

Thus, we verify K against q_Y first. Only in the equality case do we need to simulate the event $D \geqslant r_Y$. Instead of sampling D fully, we sample a Bernoulli variable $b_{T_Y} \sim \mathcal{B}(T_Y)$ where the parameter is the tail probability of the RED distribution:

$$T_Y := \mathbb{P}[D \geqslant r_Y] = 1 - \mathbb{P}[D < r_Y].$$

Using Lemma 1 (CDF matching), this probability is explicitly computable. To allow for constant-time implementation with finite randomness, we approximate T_Y by $T_Y^{(p)}$ using p bits of precision. The precision p is selected to satisfy the Rényi bound:

$$\max_{Y \in (-B_g, B_g)} R_a\left(\mathcal{B}(T_Y^{(p)}) \| \mathcal{B}(T_Y)\right) \leqslant t_R.$$

This ensures that replacing the exact comparison with a p-bit comparison preserves security. We finally obtain Algorithm 6.

Alg. 6 :: Finite precision discrete Gaussian sampler

Input: $s = p/q \in \mathbb{Q}_{>0}$, $c = p'/q' \in \mathbb{Q}$
Output: $Z \sim D_{\mathbb{Z},c,\sigma}$, $\sigma = s/\sqrt{2\ln 2}$

$\tilde{K}_0 \leftarrow (2pqq')^2$
$g \leftarrow \gcd(2q^2, p^2q')$
$K_0 \leftarrow K_0/g^2$
while TRUE **do**
  $Y \leftarrow \mathrm{DL}_{c|B_g}(\ln 2)$ // Algorithm 3
  $L_Y \leftarrow (2q^2\,|Yq' - p'| - p^2q')/g$
  $L_Y^2 = q_Y \cdot K_0 + r_Y$ // As in Equation (3)
  $K \xleftarrow{\$} \mathrm{Geom}_{|K_{\mathrm{sat}}}(1/2)$
  if $K < q_Y$ **then continue** // Early reject
  $T_Y^{(p)} := \lfloor 2^p(2^{1-r_Y/K_0} - 1)\rceil 2^{-p}$ // $\mathbb{P}[D \geqslant r_Y]$
  $b_T \xleftarrow{\$} \mathcal{B}(T_Y^{(p)})$
  if $K > q_Y \vee (K = q_Y \wedge b_T)$ **then break** // Accept
return Y

5 Masked gadgets

Though not immediately obvious, our sampler turns out to be noticeably masking-friendly. As shown later on, all the heavy computations in Algorithm 6 can be handled for free, leaving out only usual masked operations such as comparisons or equality tests. We coin this implementation the MASKAGLIA *gadget*, in honor of Georges Marsaglia (1937–2011). The gadget is parametrized by a width σ and a precision parameter ε, and outputs a Boolean sharing of an integer, which corresponding output distribution is ε-close (in Rényi divergence) to the target discrete Gaussian distribution of width σ.

Remark 3 (leakage model for samplers). We work in the t-probing model with PINI composability. The only secret term is the final masked output sample. All internal randomness is freshly sampled and independent. We thus assume that the adversary may observe up to t wires of the gadget, as well as the total number of rejection iterations (this does not depend on any secret input and therefore does not violate security).

To build MASKAGLIA from Algorithm 1, we need several elementary blocks:

- SecEq, a gadget for testing equality either between two masked values or between a masked value and a public constant.
- SecGeom, a sampling gadget which produces a masked sample G from a geometric distribution of parameter $1/2$.
- SecDLX, a sampling gadget that extends SecGeom to output a masked evaluation of any function on input G. It can be used to output a discrete Laplace sample Y follwing Algorithm 3 method, while simultaneously evaluating a family of public functions $\{f_i\}_{i=1}^n$ on $|Y - c|$.
- known primitives:
  - SecAnd ([CS20, Algorithm 2]), SecUnmask([ABC+23, Algorithm 3])
  - modified[6] SecLeq ([ABC+23, Algorithm 4]).

[6] Unlike [ABC+23], we opt not to unmask the output of the SecLeq gadget, as our sampler sometimes require a masked output. In scenarios where the comparison result can be safely made public, we perform an explicit

Alg. 7 :: Masked equality test SecEq^d_ψ

Input : Boolean sharing $\boldsymbol{x}^k$, public integer $\psi < 2^k$.
Output : Boolean sharing $\boldsymbol{b}^1$ such that $b = (x == \psi)$.

$\boldsymbol{r}^k \leftarrow \boldsymbol{x}^k \oplus \psi$
$\boldsymbol{res}^1 \leftarrow \boldsymbol{r}^0$
for $i \in \{1, \dots, k-1\}$ **do** $\boldsymbol{res}^1 \leftarrow \mathsf{SecOr}_1(\boldsymbol{res}^1, \boldsymbol{r}^i)$
return $\mathsf{SecNot}(\boldsymbol{res}^1)$

- XOR and NOT (denoted by $\neg$), which are linear operations performed sharewise and satisfy the PINI property [CS20].

We give correctness proofs for each gadget and show that they are PINI, thereby ensuring composability and t-probing security for the complete sampler. Finally, we quantify efficiency in terms of the number of SecAnd calls, since it is the only non-linear primitive.

5.1 Secure equality test

We give our secure equality test in Algorithm 7, a bitsliced-friendly variant of [CGMZ23, Algorithm 13] and prove its PINI security.

Remark 4. The SecOr gadget can be implemented using the SecAnd gadget and the $\neg$ linear operation, leveraging the fact that $\mathsf{SecOr}(\boldsymbol{a}^k, \boldsymbol{b}^k) = \neg(\mathsf{SecAnd}(\neg\boldsymbol{a}^k, \neg\boldsymbol{b}^k))$.

Amortized SecAnd cost. The $\mathsf{SecEq}_\psi\left(\boldsymbol{x}^k\right)$ gadget performs $k-1$ calls to SecOr over 1 bit, which is equivalent to $k-1$ calls to SecAnd. When fully bitsliced over a w-bit architecture, this amounts to an amortized SecAnd cost of $(k-1)/w$ SecAnd^d_w.

Proposition 2. *Algorithm 7 is PINI and, on input $\boldsymbol{x}^k$, $\psi < 2^k$, outputs a sharing of a bit b with $b = 1$ if $x = \psi$, $b = 0$ otherwise.*

The proof of Proposition 2 can be found in Section C.

5.2 Secure geometric sampling, SecGeom

The SecGeom gadget in Algorithm 9 samples (under Boolean masking) an integer $p \in \{0, \dots, k-1\}$ with the truncated Geom(1/2) law.

It relies on the classical "position of the first 1" trick. We draw a masked uniform k-bit string $x \in \{0,1\}^k$ and define p as the index of the least significant 1 bit. Then

$$\Pr[p = i] = \Pr[x[0] = \cdots = x[i-1] = 0 \ \wedge \ x[i] = 1] = 2^{-(i+1)},$$

except for the truncation event where no 1 appears among the first k bits.

To compute p in constant time, we scan x from low to high while maintaining a 1-bit masked flag $prev$ that indicates whether a 1 has already been seen. At position i we form the masked

call to SecUnmask following the gadget's execution. The aforementioned gadget can be adapted to handle two masked inputs, $\boldsymbol{x}^k$ and $\boldsymbol{y}^k$. In this case, the comparison relies on the relation $\boldsymbol{x}^k - \boldsymbol{y}^k = \boldsymbol{x}^k + \neg\boldsymbol{y}^k + 1$. The addition is implemented using a chain of 1-bit full adders, as described in [BC22, Algorithm 5]. Notably, the addition of the constant 1 is performed at no additional cost by initializing the carry-in of the first adder to 1 instead of 0.

Alg. 8 :: Select^d

Input : Public integer $x \geq 0$, Boolean sharing of a control bit $\boldsymbol{b}^1$
Output : Boolean sharing $\boldsymbol{r}^\ell$ of x if $b = 1$, else of 0, where $\ell := \lceil \log_2(x+1) \rceil$

$\ell \leftarrow \lceil \log_2(x+1) \rceil$
for $j \in \{0, \ldots, \ell - 1\}$ **do**
  if $x[j] = 1$ **then** $\boldsymbol{r}^j \leftarrow \boldsymbol{b}^1$
  else $\boldsymbol{r}^j \leftarrow \boldsymbol{0}^1$
return $\boldsymbol{r}^\ell$

Alg. 9 :: $\mathsf{SecGeom}_k^d$

Input : Public parameters $k \in \mathbb{N}$ support size
Output : Boolean sharing $\boldsymbol{r}^{k'}$ of $p \in \{0, \ldots, k-1\}$ that follows a truncated geometric distribution

$\boldsymbol{u}^k \leftarrow \mathsf{SecUnif}(k)$
$k' \leftarrow \lceil \log_2(k) \rceil$
$\boldsymbol{prev}^1 \leftarrow \neg \boldsymbol{u}^1[0]$
$\boldsymbol{r}^{k'} \leftarrow \mathsf{Select}(0, \boldsymbol{u}^1[0])$
for $i \in \{1, \ldots, k-1\}$ **do**
  $\boldsymbol{b}^1 \leftarrow \mathsf{SecAnd}_1^d(\boldsymbol{prev}^1, \boldsymbol{u}^1[i])$
  $\boldsymbol{r}^{k'} \leftarrow \boldsymbol{r}^{k'} \oplus \mathsf{Select}(i, \boldsymbol{b}^1)$
  $\boldsymbol{prev}^1 \leftarrow \boldsymbol{prev}^1 \oplus \boldsymbol{b}^1$
return $\boldsymbol{r}^{k'}$

control bit $b_i := \mathsf{AND}(prev, x[i])$, which is 1 exactly for the first 1 in x. We use b_i to conditionally add i into the masked output register for p.

To do so, we need a small helper Select primitive described in Algorithm 8: given a public integer x and a masked control bit $\boldsymbol{b}^1$, it outputs a sharing of x if $b = 1$, and a sharing of 0 otherwise. We also employ a masked uniform sampler $\mathsf{SecUnif}^d(k)$, which generates d shares of a random k-bit value.

Amortized SecAnd cost. The $\mathsf{SecGeom}_k^d$ gadget performs $k-1$ calls to SecAnd over 1 bit. When fully bitsliced over a w-bit architecture, this amounts to an amortized SecAnd cost of $(k-1)/w$ SecAnd_w^d.

Proposition 3. *Algorithm 9 is PINI. Moreover, on input $k \in \mathbb{N}$ support size, it outputs a Boolean sharing of a sample p whose distribution is the truncated geometric distribution of parameter $1/2$ over $\{0, \ldots, k-1\}$.*

The proof of that proposition is similar to the one of Proposition 4, and is provided in Section D.

5.3 Secure discrete Laplace sampling with functional evaluation, SecDLX

We introduce SecDLX in Algorithm 10, a masked adaptation of the discrete Laplace sampling procedure described in Algorithm 3. Beyond sampling the base geometric variable G as performed by the SecGeom gadget, SecDLX is designed to simultaneously evaluate a family of publicly computable functions $(f_{0,j}, f_{1,j})_{0 \leq j < n}$ on the fly. This mechanism is primarily utilized to compute the discrete Laplace sample Y,defined as $\lfloor c \rfloor + G + 1$ or $\lfloor c \rfloor - G$ depending on a secret Bernoulli bit S, and to efficiently evaluate the functions of $|Y - c|$ required for the quadratic rejection test in Gaussian sampling.

Alg. 10 :: SecDLX_k^d

Input : Public functions $(f_{0,j}, f_{1,j})_{0 \le j < n}$, public parameter $c \in \mathbb{Q}$.
Output : Boolean sharings $\boldsymbol{r_j^{s_j}}$ of $f_{b,j}(p)$, where p follows a truncated geometric$(1/2)$ over $\{0,\dots,k-1\}$ and b is secret and follows $\mathcal{B}(\frac{2^\delta}{2^\delta - 2^{1-\delta}})$ with $\delta = c - \lfloor c \rfloor$.

$\boldsymbol{u}^k \leftarrow \mathsf{SecUnif}^d(k)$
$\boldsymbol{S}^1 \leftarrow \mathsf{SecBernoulli}^d(\frac{2^\delta}{2^\delta + 2^{1-\delta}})$
$\boldsymbol{prev}^1 \leftarrow \neg \boldsymbol{u}^1[0]$
$\boldsymbol{\sigma}^1 \leftarrow \mathsf{SecAnd}_1^d(\boldsymbol{u}^1[0], \boldsymbol{S}^1)$ `// Branch selection`
for $j \in \{0,\dots,n-1\}$ **do**
  $s_j \leftarrow \max(s(f_{0,j}), s(f_{1,j}))$
  $\boldsymbol{r_j^{s_j}} \leftarrow \mathsf{Select}(f_{0,j}(0), \boldsymbol{u}^1[0] \oplus \boldsymbol{\sigma}^1) \oplus \mathsf{Select}(f_{1,j}(0), \boldsymbol{\sigma}^1)$
for $i \in \{1,\dots,k-1\}$ **do**
  $\boldsymbol{b}^1 \leftarrow \mathsf{SecAnd}_1^d(\boldsymbol{prev}^1, \boldsymbol{u}^1[i])$
  $\boldsymbol{\sigma}^1 \leftarrow \mathsf{SecAnd}_1^d(\boldsymbol{b}^1, \boldsymbol{S}^1)$
  for $j \in \{0,\dots,n-1\}$ **do**
    $\boldsymbol{t}^{s_j} \leftarrow \mathsf{Select}(f_{0,j}(i), \boldsymbol{b}^1 \oplus \boldsymbol{\sigma}^1) \oplus \mathsf{Select}(f_{1,j}(i), \boldsymbol{\sigma}^1)$
    $\boldsymbol{r_j^{s_j}} \leftarrow \boldsymbol{r_j^{s_j}} \oplus \boldsymbol{t}^{s_j}$
  $\boldsymbol{prev}^1 \leftarrow \boldsymbol{prev}^1 \oplus \boldsymbol{b}^1$
return $(\boldsymbol{r_0^{s_0}}, \dots, \boldsymbol{r_{n-1}^{s_{n-1}}})$

Evaluating functions of $|Y - c|$ presents a specific challenge because the input depends on the secret bit S, which corresponds to the sample's side relative to the center. To maintain this bit secret, SecDLX evaluates both potential function branches $(f_{0,j}, f_{1,j})$ and performs a secret selection using the control bit of the Select primitive.

To minimize computational overhead, we implement a specialized multiplexing optimization. At each bit-iteration i of the geometric sampling loop, we compute a single selection share $\boldsymbol{\sigma}^1 = \mathsf{SecAnd}_1^d(\boldsymbol{b}^1, \boldsymbol{S}^1)$, where b is the indicator bit for the position of the first 1 in the uniform randomness. This selection bit allows us to recover the complementary branch share via a simple linear operation: $\boldsymbol{b}^1 \oplus \boldsymbol{\sigma}^1 = \boldsymbol{b}^1 \wedge \neg S$.

For any function f defined over a finite domain of $\mathbb{N}$, we denote by $s(f)$ the bit-width required to represent its image. For a positive function f defined over k-bit integers, this width is $s(f) = \lceil \log(\max_{0 \le x < 2^k}(f(x))) \rceil$. In cases where f_j admits negative values, an appropriate signed encoding—such as two's complement or a biased representation—must be chosen. The gadget accordingly sizes the output registers to accommodate these encoded values during accumulation.

Amortized SecAnd cost. The SecDLX_k^d gadget performs $2(k-1)$ calls to SecAnd_1^d (excluding the cost of SecBernoulli). When fully bitsliced over a w-bit architecture, this amounts to an amortized cost of $2(k-1)/w$ SecAnd_w^d.

Proposition 4. *Algorithm 10 is PINI. Moreover, on input $(f_{b,j})_{0 \le j < n}$ and c, it outputs a Boolean masking of $f_{b,j}(p)$ where p follows a truncated geometric distribution of parameter $1/2$ and b is a secret Bernoulli bit.*

Proof. **Correctness**. Let $u \in \{0,1\}^k$ and define $p \in \{0,\dots,k-1\}$ as the index of the least significant 1 in u (i.e., $u[0] = \cdots = u[p-1] = 0$ and $u[p] = 1$). Let $S \in \{0,1\}$ be the secret Bernoulli bit. We show that Algorithm 10 outputs shares of $f_{S,j}(p)$ for each j.

Consider the scan over $i = 0,\dots,k-1$. Define the invariant after processing position i:

Alg. 11 :: $\mathsf{LaplaceMap}_c$

Input: $G \in \mathbb{N}$, public parameter $c \in \mathbb{Q}$, branch bit $b \in \{0,1\}$

1 **if** $b = 0$ **then return** $-\lfloor c \rfloor - G$
2 **else return** $\lfloor c \rfloor + G + 1$

1. $prev = 1$ iff $u[0] = \cdots = u[i] = 0$ (no 1 has been seen yet);
2. if a 1 occurred among $u[0], \ldots, u[i]$, then for all j, $r_j = f_{S,j}(p)$; otherwise $r_j = 0$.

Initialization ($i = 0$). We set $prev \leftarrow \neg u[0]$, so (1) holds. We compute $\sigma = u[0] \wedge S$. The update for r_j uses selection bits $u[0] \oplus \sigma = u[0] \wedge \neg S$ and $\sigma = u[0] \wedge S$. Thus:

$$r_j = (f_{0,j}(0) \cdot (u[0] \wedge \neg S)) \oplus (f_{1,j}(0) \cdot (u[0] \wedge S))$$

If $p = 0$, then $r_j = f_{S,j}(0)$, otherwise $r_j = 0$. Thus (2) holds.

Induction step. Assume the invariant holds for $i-1$. We compute $b \leftarrow prev \wedge u[i]$. By (1), $b = 1$ iff $p = i$. The selection logic uses $\sigma = b \wedge S$ and $b \oplus \sigma = b \wedge \neg S$. The term t_j is:

$$t_j = (f_{0,j}(i) \cdot (b \wedge \neg S)) \oplus (f_{1,j}(i) \cdot (b \wedge S))$$

If $p = i$ ($b = 1$), then $t_j = f_{S,j}(i)$. Since r_j was 0, it becomes $f_{S,j}(i)$. If $p \neq i$ ($b = 0$), then $t_j = 0$ and r_j remains unchanged, preserving (2). Finally, $prev \leftarrow prev \oplus b$, which flips $prev$ from 1 to 0 exactly when the first 1 is encountered, preserving (1).

PINI security. The security relies on the PINI composition property. The secret Bernoulli bit S and the uniform bits u_i are used as inputs to PINI gadgets (SecAnd, SecXor, SecNot). In each iteration, the indicator bit b (which is a PINI sharing) is combined with S to form the selection bit σ. Although b is reused to compute both σ and $prev$, and σ is reused across the functions f_j, all these operations are either linear (SecXor, SecNot) or constitute valid chains of PINI gadgets. Since no forbidden share-reuse occurs between non-linear operations without appropriate refreshing (guaranteed here by the fresh randomness of u_i), the gadget is PINI. □

We define the auxiliary public function of Line 11, denoted by $\mathsf{LaplaceMap}_c$, which serve to map the base geometric sample to the discrete Laplace distribution. When these functions are assigned to the initial inputs $f_{0,0}$ and $f_{1,0}$ of Algorithm 10, the first output is distributed according to $DL_c(\ln 2)$.

Particular cases of $c \in \{0, 1/2\}$. While Algorithm 10 provides a general framework for any center $c \in \mathbb{Q}$, the case $c = 0$ is most common, and the case $c = 1/2$ offers significant optimizations and is used in Hawk. The gadget's operations have two purposes:

- Sampling the discrete Laplace variable Y, defined as $\lfloor c \rfloor - G$ if $S = 0$ and $\lfloor c \rfloor + G + 1$ if $S = 1$.
- Evaluating functions based on the absolute distance $|Y - c|$, given by:

$$|Y - c| = \begin{cases} |\lfloor c \rfloor - G - c| & \text{if } S = 0 \\ |\lfloor c \rfloor + G + 1 - c| & \text{if } S = 1 \end{cases}$$

Case $c = 0$. In this scenario, the Bernoulli parameter is $1/3$. Consequently, the cost of sampling S is on average $7/3$ SecAnd_1^d. The amortized cost of the gadget in this case is then $(2(k-1)+7/3)/w$ SecAnd_w^d operations.

Case $c = 1/2$. For $c = 1/2$, the absolute distance $|Y - c| = G + 1/2$ is independent of the secret bit S. This symmetry allows the branching logic to be bypassed for all function evaluations within

Alg. 12 :: Public functions $\mathsf{Div}_{s,c}$ and $\mathsf{Thresh}_{s,c}$

Function *$Div_{s,c}$*
- **Input** : Geometric sample $G \in \mathbb{N}$, branch bit $b \in \{0,1\}$, public parameters $s = \frac{p}{q}$ and $c = \frac{p'}{q'}$
- **Output** : Integer $q_Y + 1$ where $q_Y = \lfloor L_y^2 / K_0 \rfloor$

$g \leftarrow \gcd(2q^2, p^2 q')$
$K_0 \leftarrow (2pqq'/g)^2$
if $b = 0$ **then** $Y \leftarrow \lfloor c \rfloor - G$
else $Y \leftarrow \lfloor c \rfloor + G + 1$
$L_Y \leftarrow \frac{2q^2|Yq' - p'| - p^2 q'}{g}$
$q_Y \leftarrow \lfloor L_Y^2 / K_0 \rfloor$
return $(q_Y - 1, q)$ `// Both values are needed for the strict inequality test`

Function *$Thresh_{s,c}$*
- **Input** : Geometric sample $G \in \mathbb{N}$, branch bit $b \in \{0,1\}$, precision parameter $p_U \in \mathbb{N}$, public parameters $s = \frac{p}{q}, c = \frac{p'}{q'}$, precision parameter p_U
- **Output** : Precision-adjusted threshold T_Y for the quadratic residue $L_y^2 \pmod{K_0}$

`// Precomputed constants`
$K_0 \leftarrow (2pqq'/\gcd(2q^2, p^2 q'))^2$
if $b = 0$ **then** $Y \leftarrow \lfloor c \rfloor - G$
else $Y \leftarrow \lfloor c \rfloor + G + 1$
$L_y \leftarrow \frac{2q^2|yq' - p'| - p^2 q'}{g}$
$r_Y \leftarrow L_y^2 \pmod{K_0}$
return $T_Y = \left\lfloor 2^{p_U} \cdot \left(2 - 2^{1 - r_Y/K_0}\right) \right\rceil$

the rejection loop, as they only depend on G. This reduces the loop's core cost from $2(k-1)$ to $k-1$ SecAnd_1^d calls. The discrete Laplace sample is only reconstructed once a candidate is accepted, using the relation $Y = ((S|\ldots|S) \oplus \neg G) + 1$. While this final addition requires $k-1$ SecAnd_1^d calls for carry propagation, it is performed only once per output sample rather than at each rejection step. The Bernoulli parameter is $1/2$ and the bit S can be sampled using a single uniform masked bit, bypassing the need for SecLeq calls and rendering the Bernoulli cost negligible. The total amortized cost within the loop is $(k-1)/w$ SecAnd_w^d, plus an extra $(k-1)/w$ SecAnd_w^d not affected by the rejection factor.

5.4 The MASKAGLIA Sampler

We can now finalize the implementation of our MASKAGLIA gadget, described in Algorithm 13. A key feature of this construction is the substitution of the direct acceptance condition $K_0 K + D \geqslant L_Y^2$ with the more efficient modular test described in Section 4.2. This approach avoids the explicit evaluation of the arithmetic operations within the original rejection expression and limits the bit-width of the required comparisons.

To implement this efficiently, we define two auxiliary public functions, $\mathsf{Div}_{s,c}$ and $\mathsf{Thresh}_{s,c}$ (Algorithm 12 and Line 13) which are evaluated concurrently during the sampling of the Laplace

Alg. 13 :: $\textsc{Maskaglia}^d_{s,c,\varepsilon}$

Input : Public parameters standard deviation parameter $s = p/q$, center $c = p'/q' \in Q$, precision parameter $\varepsilon \in \mathbb{R}$, Rényi parameter $a > 1$.
Output : Boolean sharing of a discrete Gaussian sample Y.

// Precomputed constants:
// p_Y, p_K, p_U are the bit sizes of the random variables Y, K, U and B_g, K_{sat} the bounds of Y, K derived from (ε, s, c, a), see Section 4.2

while TRUE **do**
  $(\boldsymbol{Y}^{p_Y}, (\boldsymbol{q_Y - 1})^{p_K}, \boldsymbol{q_Y}^{p_K}, \boldsymbol{T_Y}^{p_U}) \leftarrow \mathsf{SecDLX}^d_{B_g}((\mathsf{LaplaceMap}_c(b,\cdot))_b, (\mathsf{Div}_{s,c}(b,\cdot))_b, (\mathsf{Thresh}_{s,c}(b,\cdot))_b, c)$
  $\boldsymbol{K}^{p_K} \leftarrow \mathsf{SecGeom}^d_{K_{\text{sat}}}()$
  $\boldsymbol{b}^1_{\boldsymbol{leq}} \leftarrow \mathsf{SecLeq}^d(\boldsymbol{K}^{p_K}, (\boldsymbol{q_Y - 1})^{p_K})$
  // If $K < q_Y$, early reject
  if $\mathsf{SecUnmask}^d_1(\boldsymbol{b}^1_{\boldsymbol{leq}})$ **then**
    $\boldsymbol{b}^1_{\boldsymbol{eq}} \leftarrow \mathsf{SecEq}^d(\boldsymbol{K}^{p_K}, \boldsymbol{q_Y}^{p_K})$
    $\boldsymbol{U}^{p_U} \leftarrow \mathsf{SecUnif}^d(p_U)$
    $\boldsymbol{b}^1_{\boldsymbol{leq}} \leftarrow \mathsf{SecLeq}^d(\boldsymbol{U}^{p_U}, \boldsymbol{T_Y}^{p_U})$
    $\boldsymbol{r}^1 \leftarrow \boldsymbol{b}^1_{\boldsymbol{eq}} \oplus \mathsf{SecAnd}^d_1(\neg \boldsymbol{b}^1_{\boldsymbol{eq}}, \boldsymbol{b}^1_{\boldsymbol{leq}})$
    // Accept: $K > q_Y \vee (K = q_Y \wedge (D > L_Y^2 \pmod{K_0}))$
    if $\mathsf{SecUnmask}(\boldsymbol{r}^1)$ **then break**
return $\boldsymbol{Y}^{p_Y}$

candidate via the SecDLX gadget. On top of the public parameters, these functions are parameterized by a secret control bit b that determines the Y's position relative to the center. By internally computing the absolute distance $|Y - c|$, these functions allow the gadget to determine the necessary algebraic terms (the quotient $q_Y = \lfloor L_Y^2 / K_0 \rfloor$ and the threshold T_Y corresponding to $r_Y = L_Y^2 \pmod{K_0}$) on the fly during the sampling process.

Remark 5. While the $\mathsf{Thresh}_{s,c}$ support can be lengthy, there is a unique threshold value for each possible quadratic residue. Depending on implementation constraints, these thresholds can be precomputed and stored, or calculated at runtime.

Amortized SecAnd cost. Let B_g, K_{sat} be the bounds of random variables Y, K defined in Section 4.2. The computational cost of $\textsc{Maskaglia}^d_{s,c,\varepsilon}$ can be broken down into the following components:

- The $\mathsf{SecDLX}^d_{B_g}$ gadget requires $2(B_g - 1)$ calls to SecAnd^d_1, in addition to the cost associated with SecBernoulli (see Section 5.3).
- The $\mathsf{SecGeom}^d_{K_{\text{sat}}}$ gadget requires $K_{\text{sat}} - 1$ calls to SecAnd^d_1.
- The two gadgets $\mathsf{SecLeq}^d_{p_K}$ and $\mathsf{SecLeq}^d_{p_U}$ require $\lceil \log(K_{\text{sat}}) \rceil - 1 = p_K - 1$ and $p_U - 1$ calls to SecAnd^d_1, respectively.
- The $\mathsf{SecEq}^d_{p_K}$ gadget accounts for $p_K - 1$ calls to SecAnd^d_1.

This base cost must be scaled by the rejection factor r_f derived in Corollary 1. The performance gains from early rejection can be quantified empirically; the computations are included in the script provided as supplementary material.

In a fully bitsliced implementation targeting a w-bit architecture, the total cost of the sampler results in an amortized cost of:

$$\frac{r_f \times (2B_g + K_{\text{sat}} + 2p_K + p_U - 5)}{w} \text{ calls to } \mathsf{SecAnd}_w^d.$$

Theorem 2. *Let σ be a standard deviation such that $s = \sigma/2\ln 2 = \frac{p}{q} \in \mathbb{Q}$, $c = \frac{p'}{q'} \in \mathbb{Q}$ be a center, $\varepsilon > 0$ a target precision. Then Algorithm 13 is PINI. Moreover, on input s, c, ε, a, it outputs is a Boolean sharing of a sample Y following a distribution $\tilde{D}_{\mathbb{Z},c,\sigma}$ such that $R_a(\tilde{\mathcal{D}} \| \mathcal{D}_{\mathbb{Z},c,\sigma}) \leqslant \varepsilon$, where $D_{\mathbb{Z},c,\sigma}$ is the target discrete Gaussian distribution.*

Proof. **Correctness**. We establish that Algorithm 13 is a masked, finite-precision equivalent of Algorithm 6. Both algorithms follow analogous distributions for the candidate Y and the variable K, subject to precision constraints. Consequently, the quotient q_Y remains consistent across both implementations.

The masked threshold $\mathbf{T_Y}$ is constructed to satisfy $\mathbf{T_Y} = \lfloor 2^p(1 - T_Y^{(p)}) \rfloor$. For a uniform integer $U \leftarrow \mathcal{U}(\{0, \ldots, 2^p - 1\})$, the probability of acceptance in the equality case satisfies:

$$\mathbb{P}(U \leqslant \mathbf{T_Y}) = \mathbb{P}(\mathcal{B}(1 - T_Y^{(p)}) = 1) = 1 - T_Y^{(p)}.$$

This ensures that the rejection criteria are statistically equivalent to those of the ideal sampler. As detailed in Section 4.2, the precision parameters are chosen such that the instantiated distribution $\tilde{\mathcal{D}}$ satisfies $R_a(\tilde{\mathcal{D}} \| \mathcal{D}_{\mathbb{Z},c,\sigma}) \leqslant \varepsilon$ for all $a \geqslant 2$, where $\mathcal{D}_{\mathbb{Z},c,\sigma}$ denotes the ideal target discrete Gaussian distribution.

PINI Security. The security of the gadget follows from the PINI composition property. The gadget unmasks the early rejection bit b_{leq} and the final acceptance bit r, which only leaks information on rejected samples independent to the final output. Thus, the complete MASKAGLIA gadget is PINI-secure. □

6 Performance evaluation and comparison

Cost framework. In a masked implementation, efficiency is primarily governed by the number of non-linear (quadratic) operations. While this is particularly evident in higher-order schemes, it remains a critical factor even at order 2. Even with access to a hardware True Random Number Generator (TRNG), randomness generation typically dominates the total execution time of a gadget. Under the realistic assumption that generating a single random bit requires slightly more than one clock cycle, the cost of randomness at order 2 can exceed that of linear operations by more than a factor of ten. Since PINI-secure gadgets incorporate automatic mask refreshing, randomness consumption (and by extension, the overall runtime) is concentrated within the SecAnd operations. Consequently, the SecAnd gate count serves as the most relevant metric for assessing and comparing computational efficiency.

A second key point is the impact of *bitslicing* on the cost of masked operations. Since bitslicing fully leverage the native parallelism of 32-bit (or larger) micro-controllers, the standard metric for Boolean-masked algorithms must shift from raw gate count to an *amortized* cost. For instance, on a 32-bit architecture, a bit-level sampler requiring 32 bitwise sequential SecAnd_1 gates can process a full batch of 32 samples in parallel, resulting in an amortized cost of only 1 SecAnd_{32} per sample. Conversely, a word-level sampler requiring only 5 SecAnd_{32} per sample cannot benefit from such packing, as each gate already occupies the full register width. Over a batch of 32 samples, the bit-level approach is thus five times more efficient (32 vs. 160 SecAnd_{32}). In the context of Gaussian sampling for lattice-based signature schemes, the large number of samples required ($n \gg 32$) ensures that the native parallelism of a 32-bit architecture can be fully leveraged.

Table 2. Expected number of SecAnd gates per sample, comparing samplers for Hawk-512 and Hawk-1024 across parameter choices $c \in \{0, 0.5\}$ on a 32-bit architecture. For bitsliced rows, counts correspond to the number of 1-bit SecAnd gates divided by architecture bitsize. Ratios (in parentheses) are relative to this work.

	Hawk-{512, 1024}	
Sampler	$c = 0$	$c = 0.5$
Gérard–Rossi [GR19]	195 (× **40.1**)	195 (× **48.8**)
Eid et al. [EGRZ26]	87 (× **17.9**)	87 (× **21.7**)
Gérard–Rossi [GR19] (bitsliced)	31.7 (× **6.5**)	31.7 (× **7.9**)
Eid et al. [EGRZ26] (bitsliced)	19.5 (× **4.0**)	19.5 (× **4.9**)
This work	4.9	4.0

6.1 Maskaglia parameters

The efficiency of the masked Maskaglia sampler depends on the parameters s, c, and the target Rényi divergence t_R. The first two determine the distributions and constants used throughout the sampler, while t_R bounds the sampling domains of intermediate random variables (see Section 3.2). To simplify the evaluation of the number of SecAnd gates, we provide as supplementary material a Python script that, given (s, c, t_R), computes the bounds of Section 3.2 and estimates the expected number of SecAnd gates for the corresponding instantiation.

6.2 Comparisons

We evaluate our sampler on Hawk-512 and Hawk-1024 parameters. Competing methods' implementation do not leverage bitsliced operations and use suboptimal adders algorithm for their SecLeq, which is the most expensive operation. To ensure a fair comparison, we also consider the cost of variants of these implementations assuming they use optimized bitsliced operations. To estimate the bitsliced cost of the Gérard–Rossi [GR19] and Eid et al. [EGRZ26] gadgets for Hawk, we replace their comparison gadget, which relies on a Kogge–Stone adder and therefore does not benefit from bitslicing, by ours, and we count only the cost of comparisons. We target a Rényi divergence $t_R = 1 + 2^{-76}$ at order $a = 256$, corresponding to a loss of at most one bit of security.

The work of Eid et al. [EGRZ26] uses a binary tree representation of the CDT. The tree is traversed in a masked fashion by performing masked comparisons with an on-the-fly updated pivot value. For Hawk-512 and Hawk-1024, these comparisons are performed over 78 bits. The overall cost of [EGRZ26] is computed as the cost of:

- 4 masked comparisons over 78 bits,
- 4 SecAnd over 78 bit to update the pivot value.

Similar estimates are carried out for [GR19]. For completeness, we also provide a detailed cost comparison in terms of randomness and linear gates in Section G.

Remark 6 (boolean-to-arithmetic conversion). Both our sampler and the sampler of [EGRZ26] output Gaussian samples in Boolean masked form. In most practical deployments (in particular for lattice-based schemes like Hawk) these values will ultimately be converted to arithmetic masking. While standard techniques exist (e.g., [BC22, CGMZ22]), we exclude this step from the comparison, since it incurs the same overhead for both approaches.

References

ABC+23. Melissa Azouaoui, Olivier Bronchain, Gaëtan Cassiers, Clément Hoffmann, Yulia Kuzovkova, Joost Renes, Tobias Schneider, Markus Schönauer, François-Xavier Standaert, and Christine van Vredendaal. Protecting Dilithium against leakage revisited sensitivity analysis and improved implementations. 2023(4):58–79, 2023.

AS72. Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. Dover, 1972.

BBD+24. Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK. Technical report, National Institute of Standards and Technology, 2024. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures.

BBE+19. Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. pages 2147–2164, 2019.

BC22. Olivier Bronchain and Gaëtan Cassiers. Bitslicing arithmetic/boolean masking conversions for fun and profit with application to lattice-based KEMs. 2022(4):553–588, 2022.

BHLY16. Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. pages 323–345, 2016.

Bih97. Eli Biham. A fast new DES implementation in software. pages 260–272, 1997.

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. pages 149–164, 2025.

BPT+24. Pierre-Augustin Berthet, Justine Paillet, Cédric Tavernier, Lilian Bossuet, and Brice Colombier. Masked computation of the floor function and its application to the FALCON signature. 1(4):9, 2024.

BT25. Pierre-Augustin Berthet and Cédric Tavernier. Improving the masked division for the FALCON signature. Cryptology ePrint Archive, Report 2025/628, 2025.

But61. J. C. Butcher. Random sampling from the normal distribution. *The Computer Journal*, (3):251–253, 1961.

CC24. Keng-Yu Chen and Jiun-Peng Chen. Masking floating-point number multiplication and addition of falcon first- and higher-order implementations and evaluations. 2024(2):276–303, 2024.

CCD+24. Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. HAETAE: Shorter lattice-based fiat-shamir signatures. 2024(3):25–75, 2024.

CGMZ22. Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order table-based conversion algorithms and masking lattice-based encryption. 2022(2):1–40, 2022.

CGMZ23. Jean-Sébastien Coron, François Gérard, Simon Montoya, and Rina Zeitoun. High-order polynomial comparison and masking lattice-based encryption. 2023(1):153–192, 2023.

CPS+20. Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. ModFalcon: Compact signatures based on module-NTRU lattices. pages 853–866, 2020.

CS20. Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Transactions on Information Forensics and Security*, 15:2542–2555, 2020.

DDLL13. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. pages 40–56, 2013.

Dev86. Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

DFW20. Yusong Du, Baoying Fan, and Baodian Wei. Arbitrary-centered discrete gaussian sampling over the integers. pages 391–407, 2020.

DLP14. Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. pages 22–41, 2014.

dPKPR24. Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. pages 409–444, 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. pages 222–253, 2022.

EFGT17. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. pages 1857–1874, 2017.

EGRZ26. Elie Eid, Aurélien Greuet, Nathan Reboud, and Rina Zeitoun. Efficient high-order masking of FrodoKEMs CDT-based Gaussian sampler. *IACR TCHES*, 2026(3), 2026.

EKT24. Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors. pages 387–424, 2024.

FKT+20. Pierre-Alain Fouque, Paul Kirchner, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Key recovery from Gram-Schmidt norm leakage in hash-and-sign signatures over NTRU lattices. pages 34–63, 2020.

GG25. François Gérard and Morgane Guerreau. High-order and cortex-M4 first-order implementations of masked FrodoKEM. Cryptology ePrint Archive, Report 2025/1065, 2025.

GLSV15. Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. LS-designs: Bitslice encryption for efficient masked software implementations. pages 18–37, 2015.

GMRR22. Morgane Guerreau, Ange Martinelli, Thomas Ricosset, and Mélissa Rossi. The hidden parallelepiped is back again: Power analysis attacks on falcon. 2022(3):141–164, 2022.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. pages 197–206, 2008.

GR19. François Gérard and Mélissa Rossi. An efficient and provable masked implementation of qTESLA. In Sonia Belaïd and Tim Güneysu, editors, *CARDIS 2019*, volume 11833 of *LNCS*, pages 74–91. Springer, 2019.

GR24. Morgane Guerreau and Mélissa Rossi. A not so discrete sampler: Power analysis attacks on HAWK signature scheme. 2024(4):156–178, 2024.

HLS18. Andreas Hülsing, Tanja Lange, and Kit Smeets. Rounded Gaussians - fast and secure constant-time sampling for lattice-based crypto. pages 728–757, 2018.

HPRR20. James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous gaussian sampling: From inception to implementation. pages 53–71, 2020.

HPT25. Calvin Abou Haidar, Quentin Payet, and Mehdi Tibouchi. Crowhammer: Full key recovery attack on falcon with a single rowhammer bit flip. pages 103–135, 2025.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. pages 463–481, 2003.

KA21. Emre Karabulut and Aydin Aysu. FALCON down: Breaking FALCON post-quantum signature scheme through side-channel attacks. In *DAC 2021*, pages 691–696. IEEE, 2021.

KLS+23. Markus Krausz, Georg Land, Florian Stolz, Dennis Naujoks, Jan Richter-Brockmann, Tim Güneysu, and Lucie Kogelheide. To extend or not to extend: Agile masking instructions for PQC. Cryptology ePrint Archive, Report 2023/1287, 2023.

LDK+22. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

LZY+25. Xiuhan Lin, Shiduo Zhang, Yang Yu, Weijia Wang, Qidi You, Ximing Xu, and Xiaoyun Wang. Thorough power analysis on falcon gaussian samplers and practical countermeasure. pages 229–258, 2025.

Mar63. George Marsaglia. Generating a variable from the tail of the normal distribution. Technical Report 322, Boeing Scientific Research Laboratories, September 1963.

MKK+23. Soundes Marzougui, Ievgen Kabin, Juliane Krämer, Thomas Aulbach, and Jean-Pierre Seifert. On the feasibility of single-trace attacks on the gaussian sampler using a CDT. pages 149–169, 2023.

MW17. Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. pages 455–485, 2017.

MWG+22. Soundes Marzougui, Nils Wisiol, Patrick Gersch, Juliane Krämer, and Jean-Pierre Seifert. Machine-learning side-channel attacks on the GALACTICS constant-time implementation of BLISS. In *ARES 2022*, pages 34:1–34:11. ACM, 2022.

NAB+20. Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. 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.

PBY17. Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. pages 1843–1855, 2017.
Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. pages 80–97, 2010.
PFH+22. 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, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.
Pre17. Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. pages 347–374, 2017.
Pre23. Thomas Prest. A key-recovery attack against mitaka in the t-probing model. pages 205–220, 2023.
SD25. Vladimir Sarde and Nicolas Debande. Improvement of side-channel attacks on mitaka. Cryptology ePrint Archive, Report 2025/873, 2025.
WGS22. Nils Wisiol, Patrick Gersch, and Jean-Pierre Seifert. Cycle-accurate power side-channel analysis using the ChipWhisperer: A case study on Gaussian sampling. In Ileana Buhan and Tobias Schneider, editors, *CARDIS 2022*, volume 13820 of *LNCS*, pages 205–224. Springer, 2022.
ZLYW23. Shiduo Zhang, Xiuhan Lin, Yang Yu, and Weijia Wang. Improved power analysis attacks on falcon. pages 565–595, 2023.
ZSS20a. Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. COSAC: COmpact and scalable arbitrary-centered discrete gaussian sampling over integers. pages 284–303, 2020.
ZSS20b. Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. FACCT: FAst, Compact, and Constant-Time discrete Gaussian sampler over integers. *IEEE Trans. Computers*, 69(1):126–137, 2020.

A Proof of Corollary 1

Proof. Both correctness claims follow from Lemma 2. For the continuous acceptance probability:

$$C_{\mathrm{Lap}} = \int_{\mathbb{R}} e^{-\lambda|t-c|}\,dt = \frac{2}{\lambda}, \qquad C_{\mathrm{G}} = \int_{\mathbb{R}} e^{-(\lambda/s^2)(t-c)^2}\,dt = s\sqrt{\frac{\pi}{\lambda}}.$$

The discrete formula is the same specialization with counting measure. For the lower bound, let $\delta(c) = \mathrm{dist}(c,\mathbb{Z}) \in [0,\frac{1}{2}]$ so that there exists $z_0 \in \mathbb{Z}$ with $|z_0 - c| = \delta(c)$. Hence

$$\sum_{z\in\mathbb{Z}} e^{-(\lambda/s^2)(z-c)^2} \;\geq\; e^{-(\lambda/s^2)\delta(c)^2} \;\geq\; e^{-\lambda/(4s^2)}.$$

Also, $\sum_{z\in\mathbb{Z}} e^{-\lambda|z-c|} \;\leq\; \sum_{z\in\mathbb{Z}} e^{-\lambda|z|} \;=\; 1 + 2\sum_{k\geq 1} e^{-\lambda k} \;=\; \frac{1+e^{-\lambda}}{1-e^{-\lambda}} \;=\; \coth\!\left(\frac{\lambda}{2}\right)$, giving the bound. □

B Counting of the residue set

Throughout let $s = p/q \in \mathbb{Q}_{>0}$ and $c = p'/q' \in \mathbb{Q}$ with $\gcd(p,q) = \gcd(p',q') = 1$ and $q, q' > 0$. Define

$$g := \gcd(2q^2,\, p^2q'), \qquad A := \frac{2q^2}{g}, \qquad B := \frac{p^2q'}{g}, \qquad n := \frac{(2pqq')^2}{g^2}.$$

For $y \in \mathbb{Z}$ set

$$L_y := A\lfloor yq' - p' \rceil - B, \qquad R_y := L_y^2 \bmod n,$$

and

$$\mathcal{R}_{s,c} := \{R_y :\; y \in \mathbb{Z}\}, \qquad \mathcal{T}_{s,c} := \mathcal{R}_{s,c} \cup \{0, n\}.$$

Finally set

$$h := Aq' = \frac{2q^2q'}{g}, \qquad M := \frac{n}{h} = \frac{2p^2q'}{g} \in \mathbb{Z}_{>0},$$

and let

$$P := \{\ell \text{ odd prime} : \ \ell \mid p,\ \ell \nmid q'\}.$$

With the above notation, $A, B, h, M, n \in \mathbb{Z}_{>0}$ and $n = hM$. By definition $g \mid 2q^2$ and $g \mid p^2q'$, hence $A, B \in \mathbb{Z}_{>0}$ and $h = Aq' \in \mathbb{Z}_{>0}$. Also $M = \frac{2p^2q'}{g} \in \mathbb{Z}_{>0}$. Finally

$$hM = \frac{2q^2q'}{g} \cdot \frac{2p^2q'}{g} = \frac{4p^2q^2q'^2}{g^2} = \frac{(2pqq')^2}{g^2} = n,$$

so $n = hM \in \mathbb{Z}_{>0}$.

Lemma 5. *Let $n = hM$ with $h, M \in \mathbb{Z}_{>0}$ and let $C = a + h\mathbb{Z}/n\mathbb{Z}$ be a coset of $h\mathbb{Z}/n\mathbb{Z}$. Define*

$$f_M : \mathbb{Z}/M\mathbb{Z} \to \mathbb{Z}/M\mathbb{Z}, \qquad f_M(k) := hk^2 + 2ak \pmod{M}.$$

Then $C^{(2)} := \{x^2 \bmod n : \ x \in C\}$ satisfies $|C^{(2)}| = |\mathrm{Im}(f_M)|$. Moreover, if $d \mid M$ and $\gcd(d, h) = 1$, then

$$|C^{(2)}| \le \frac{M}{d}\, Q(d), \qquad Q(d) := \left|\{u^2 \bmod d : \ u \in \mathbb{Z}\}\right|.$$

Proof. Write $n = hM$. Each element of C is uniquely of the form $a + hk$ with $k \in \mathbb{Z}/M\mathbb{Z}$. For such k,

$$(a + hk)^2 \equiv a^2 + h\,(hk^2 + 2ak) \pmod{hM}.$$

Let $\iota : \mathbb{Z}/M\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ be the injective map $\iota(u) = hu \bmod n$; injectivity follows since $hu \equiv 0 \pmod{hM}$ implies $M \mid u$. Thus the map $k \mapsto (a + hk)^2 \bmod n$ has image size exactly $|\mathrm{Im}(f_M)|$, proving the first claim.

Now let $d \mid M$ with $\gcd(d, h) = 1$. Let $\pi : \mathbb{Z}/M\mathbb{Z} \to \mathbb{Z}/d\mathbb{Z}$ and $\rho : \mathbb{Z}/M\mathbb{Z} \to \mathbb{Z}/d\mathbb{Z}$ be reduction maps on codomain and domain, respectively. Define

$$f_d : \mathbb{Z}/d\mathbb{Z} \to \mathbb{Z}/d\mathbb{Z}, \qquad f_d(x) := hx^2 + 2ax \pmod{d}.$$

Since $d \mid M$, one has $\pi \circ f_M = f_d \circ \rho$. Hence every value in $\mathrm{Im}(f_d)$ has at most M/d preimages under f_M, because each fiber of ρ has size M/d. Therefore

$$|\mathrm{Im}(f_M)| \le \frac{M}{d}\, |\mathrm{Im}(f_d)|. \tag{4}$$

Since $\gcd(d, h) = 1$, the element h is invertible modulo d, so completing the square gives

$$f_d(x) \equiv h\left(x + \frac{a}{h}\right)^2 - \frac{a^2}{h} \pmod{d}.$$

Thus $\mathrm{Im}(f_d)$ is an affine translate of $h \cdot \{u^2 \bmod d\}$, hence $|\mathrm{Im}(f_d)| = Q(d)$. Combining with (4) yields $|C^{(2)}| = |\mathrm{Im}(f_M)| \le \frac{M}{d} Q(d)$. □

Lemma 6 (Quadratic residues modulo odd prime powers). *Let ℓ be an odd prime and $t \ge 1$. Then*

$$\frac{Q(\ell^{2t})}{\ell^{2t}} \le \frac{1}{2}.$$

Proof. We count quadratic residues modulo ℓ^{2t}.

A residue class $x \bmod \ell^{2t}$ is a square if and only if either $x \equiv 0$, or $v_\ell(x) = 2j$ is even with $0 \le j \le t - 1$ and the unit part of x is a square in $(\mathbb{Z}/\ell^{2t-2j}\mathbb{Z})^\times$. Indeed, if $x \equiv y^2 \pmod{\ell^{2t}}$ and $x \not\equiv 0$, writing $y = \ell^j u$ with u a unit yields $x \equiv \ell^{2j} u^2$ and u^2 is a unit square modulo ℓ^{2t-2j}; conversely, if $x \equiv \ell^{2j} u^2$ with u a unit modulo ℓ^{2t-j} then x is represented by $(\ell^j u)^2$.

For $m \geq 1$, the group $(\mathbb{Z}/\ell^m\mathbb{Z})^\times$ is cyclic of even order $\varphi(\ell^m)$, hence the subgroup of squares has index 2 and contains exactly $\varphi(\ell^m)/2$ elements. Therefore

$$Q(\ell^{2t}) = 1 + \sum_{j=0}^{t-1} \frac{\varphi(\ell^{2t-2j})}{2} = 1 + \frac{\ell-1}{2}\big(\ell^{2t-1} + \ell^{2t-3} + \cdots + \ell\big).$$

Using $\ell^{2t-1} + \ell^{2t-3} + \cdots + \ell = \ell(\ell^{2t}-1)/(\ell^2-1)$ gives

$$Q(\ell^{2t}) = 1 + \frac{1}{2} \cdot \frac{\ell(\ell^{2t}-1)}{\ell+1} = \frac{\ell^{2t+1}+\ell+2}{2(\ell+1)}.$$

Since $t \geq 1$ and $\ell \geq 3$, we have $\ell^{2t} \geq \ell + 2$, hence

$$Q(\ell^{2t}) \leq \frac{\ell^{2t+1}+\ell^{2t}}{2(\ell+1)} = \frac{\ell^{2t}}{2},$$

which yields the claim. □

Proposition 5 (Counting bound). *With the above notation,* $|\mathcal{T}_{s,c}| \leq 2M \cdot 2^{-|P|} + 2$.

Proof. **Step 1: reduction of** $\{L_y \bmod n\}$ **to at most two cosets.** Let $r \in \{0,1,\ldots,q'-1\}$ be the standard residue of p' modulo q'. For any $y \in \mathbb{Z}$ one has $yq'-p' \equiv -p' \pmod{q'}$, hence $|yq'-p'| \equiv r$ or $|yq'-p'| \equiv q'-r \pmod{q'}$. Conversely, if $t \geq 0$ with $t \equiv r \pmod{q'}$, then $y = (p'-t)/q' \in \mathbb{Z}$ and $|yq'-p'| = t$; if $t \equiv q'-r \pmod{q'}$, then $y = (p'+t)/q' \in \mathbb{Z}$ and $|yq'-p'| = t$. Thus the set of residues $L_y \bmod n$ is contained in the union of the (at most) two cosets of $h\mathbb{Z}/n\mathbb{Z}$:

$$C_1 := (Ar - B) + h\mathbb{Z}/n\mathbb{Z}, \qquad C_2 := (A(q'-r) - B) + h\mathbb{Z}/n\mathbb{Z},$$

since increasing t by q' increases $L = At - B$ by $Aq' = h$. If $q' \mid 2$ (equivalently $q' \in \{1,2\}$), then $r \equiv q'-r \pmod{q'}$ and hence $C_1 = C_2$; otherwise they are distinct cosets. Writing $C^{(2)} := \{x^2 \bmod n : x \in C\}$, we have

$$\mathcal{R}_{s,c} \subseteq C_1^{(2)} \cup C_2^{(2)} \qquad \Longrightarrow \qquad |\mathcal{R}_{s,c}| \leq |C_1^{(2)}| + |C_2^{(2)}|. \tag{5}$$

Let $C = a + h\mathbb{Z}/n\mathbb{Z}$ be any coset. Applying Lemma 5 gives: for any $d \mid M$ with $\gcd(d,h) = 1$,

$$|C^{(2)}| \leq \frac{M}{d}\, Q(d). \tag{6}$$

Step 2: choosing d **from** P. Define

$$d := \prod_{\ell \in P} \ell^{2v_\ell(p)}.$$

We claim $d \mid M$ and $\gcd(d,h) = 1$. Fix $\ell \in P$. Then $\ell \mid p$ and $\ell \nmid q'$ by definition. Also $\gcd(p,q) = 1$ implies $\ell \nmid q$, hence $\ell \nmid 2q^2$. Therefore $\ell \nmid g = \gcd(2q^2, p^2q')$, and so

$$v_\ell(M) = v_\ell\left(\frac{2p^2q'}{g}\right) = 2v_\ell(p),$$

which implies $\ell^{2v_\ell(p)} \mid M$. Hence $d \mid M$. Moreover

$$h = \frac{2q^2q'}{g}$$

is not divisible by ℓ (since $\ell \nmid q$, $\ell \nmid q'$, and $\ell \nmid g$), so indeed $\gcd(d,h) = 1$.

Applying (6) to each of C_1 and C_2 with this choice of d and using (5) yields

$$|\mathcal{R}_{s,c}| \le 2 \cdot \frac{M}{d} Q(d) = 2M \cdot \frac{Q(d)}{d}. \tag{7}$$

(When $C_1 = C_2$, the factor 2 may be omitted; see Remark 7 below.)

Step 3: bounding $Q(d)/d$. By the Chinese remainder theorem, $Q(\cdot)$ is multiplicative on coprime moduli; hence

$$\frac{Q(d)}{d} = \prod_{\ell \in P} \frac{Q(\ell^{2v_\ell(p)})}{\ell^{2v_\ell(p)}}.$$

By Lemma 6, each factor is at most $1/2$, so

$$\frac{Q(d)}{d} \le \prod_{\ell \in P} \frac{1}{2} = 2^{-|P|}.$$

Substituting into (7) gives

$$|\mathcal{R}_{s,c}| \le 2M \cdot 2^{-|P|}.$$

Finally, since $\mathcal{T}_{s,c} = \mathcal{R}_{s,c} \cup \{0, n\}$, we obtain

$$|\mathcal{T}_{s,c}| \le |\mathcal{R}_{s,c}| + 2 \le 2M \cdot 2^{-|P|} + 2.$$

One-coset improvement under $g \mid 4p^2$.

Corollary 3. *If $g \mid 4p^2$, then $\mathcal{R}_{s,c}$ is contained in a single coset of the subgroup $h\mathbb{Z}/n\mathbb{Z}$, and in particular*

$$|\mathcal{T}_{s,c}| \;\le\; \min\big(2M \cdot 2^{-|P|},\, M\big) + 2.$$

Proof. Let $a_1 := Ar - B$ and $a_2 := A(q' - r) - B$, so $C_i = a_i + h\mathbb{Z}/n\mathbb{Z}$. Since every $x \in C_i$ satisfies $x \equiv a_i \pmod h$, we have $x^2 \equiv a_i^2 \pmod h$, and therefore

$$C_i^{(2)} \subseteq a_i^2 + h\mathbb{Z}/n\mathbb{Z} \qquad (i = 1, 2).$$

We claim that $a_1^2 \equiv a_2^2 \pmod h$ if and only if $g \mid 4p^2$.

Indeed, modulo $h = Aq'$ we have $a_2 \equiv -Ar - B$, hence

$$a_2^2 - a_1^2 \equiv (Ar + B)^2 - (Ar - B)^2 = 4ArB \pmod h.$$

Thus $a_1^2 \equiv a_2^2 \pmod h$ if and only if $Aq' \mid 4ArB$, i.e. $q' \mid 4rB$. If $q' = 1$ this holds trivially; if $q' \ge 2$, then $\gcd(r, q') = 1$ (since $r \equiv p' \pmod{q'}$ and $\gcd(p', q') = 1$), so $q' \mid 4rB$ is equivalent to $q' \mid 4B$. Using $B = p^2 q'/g \in \mathbb{Z}$, we have

$$q' \mid 4B \iff \frac{4B}{q'} = \frac{4p^2}{g} \in \mathbb{Z} \iff g \mid 4p^2.$$

This proves the claim.

Assume now that $g \mid 4p^2$, hence $a_1^2 \equiv a_2^2 \pmod h$, so the cosets $a_1^2 + h\mathbb{Z}/n\mathbb{Z}$ and $a_2^2 + h\mathbb{Z}/n\mathbb{Z}$ coincide in $\mathbb{Z}/n\mathbb{Z}$. Therefore $\mathcal{R}_{s,c} \subseteq a_1^2 + h\mathbb{Z}/n\mathbb{Z}$, a single coset of size M (since $n = hM$), and hence $|\mathcal{R}_{s,c}| \le M$. Combining with the bound from Step 4 gives $|\mathcal{R}_{s,c}| \le \min\big(2M \cdot 2^{-|P|}, M\big)$, and adjoining $\{0, n\}$ yields

$$|\mathcal{T}_{s,c}| \le \min\big(2M \cdot 2^{-|P|}, M\big) + 2.$$

□

Remark 7. If $q' \in \{1,2\}$, then $r \equiv q' - r \pmod{q'}$ and hence $C_1 = C_2$ in Step 1. In this case

$$|\mathcal{R}_{s,c}| \leq \frac{M}{d} Q(d)$$

with the same choice of $d = \prod_{\ell \in P} \ell^{2v_\ell(p)}$, and therefore

$$|\mathcal{T}_{s,c}| \leq M \cdot 2^{-|P|} + 2.$$

□

C Proof of Proposition 2

Proof. **Correctness.** Let $r := x \oplus \psi$. The loop iteratively computes the disjunction of the bits of r: after iteration i we have $res = r[0] \vee r[1] \vee \cdots \vee r[i]$. Hence after the loop, $res = 0$ iff $r = 0$, i.e., iff $x = \psi$; finally SecNot(res) flips this bit and outputs 1 exactly when $x = \psi$.

PINI security. The circuit is a composition of PINI gadgets (SecXor, SecOr, SecNot). Moreover, in the chain of SecOr calls, each non-linear gate consumes a fresh input bit $\boldsymbol{r}^i$ (each used once) together with the running accumulator. By the standard PINI composition criterion [CS20], this prevents forbidden share-reuse paths across non-linear gadgets, so the whole SecEq gadget is PINI. □

D Proof of Proposition 3

Proof. **Correctness.** Let $u \in \{0,1\}^k$ and define $p \in \{0,\dots,k-1\}$ as the index of the least significant 1 in u when it exists (i.e., $u[0] = \cdots = u[p-1] = 0$ and $u[p] = 1$). We show that Algorithm 9 outputs sharings of p.

Consider the scan over $i = 0,\dots,k-1$. Define the invariant after processing position i:

1. $prev = 1$ iff $u[0] = \cdots = u[i] = 0$ (no 1 has been seen yet);
2. if a 1 occurred among $u[0],\dots,u[i]$, then $r = p$, otherwise $r = 0$.

Initialization ($i = 0$). We set $prev \leftarrow \neg(u[0])$, so (1) holds. Also, we XOR r with Select$(0, u[0])$, hence $r = 0$ in both cases. Thus (2) holds.

Induction step. Assume the invariant holds after processing $i-1$ with $i \geq 1$. We compute $b_i := prev \wedge u[i]$. By (1), $b_i = 1$ iff $u[0] = \cdots = u[i-1] = 0$ and $u[i] = 1$, i.e., iff $p = i$. Therefore, XORing r with Select(i, b_i) sets r to i exactly in the case $p = i$, and leaves it unchanged otherwise. Finally, we update $prev \leftarrow prev \oplus b_i$, which flips $prev$ from 1 to 0 exactly when the first 1 is encountered, preserving (1). This establishes the invariant.

After processing $k-1$, if i contains a 1 then (2) gives $r = p$.

PINI security. The circuit is a composition of PINI gadgets (SecAnd, SecXor, SecNot) and linear operations involving public constants (Select). Each iteration uses a fresh randomness bit $\boldsymbol{u}^1[i]$ in exactly one non-linear call SecAnd$_1^d(\boldsymbol{prev}^1, \boldsymbol{u}^1[i])$; all subsequent updates in the iteration are linear. By the standard PINI composition criterion [CS20], this prevents forbidden share-reuse patterns across non-linear gadgets, hence Algorithm 9 is PINI. □

E Secure Bernoulli sampling

Alg. 14 :: Bernoulli Sampling SecBernoulli(p/q)

Input : Probability $p/q \in \mathbb{Q}$ such that $\gcd(p,q) = 1$ and $0 < p < q$
Output : Boolean sharing $\boldsymbol{r}^1$ of $r \in \{0,1\}$ such that $\Pr(r = 1) = p/q$

$k \leftarrow \lceil \log_2(q) \rceil$
repeat
  $\boldsymbol{U}^k \leftarrow \mathsf{SecUnif}^d(k)$
  $\boldsymbol{b}^1 \leftarrow \mathsf{SecLeq}^d(\boldsymbol{U}^k, q-1)$
  $b \leftarrow \mathsf{SecUnmask}_1^d(\boldsymbol{b}^1)$
until b
// U is uniform over $[0, q-1]$
$\boldsymbol{r}^1 \leftarrow \mathsf{SecLeq}^d(\boldsymbol{U}^k, p-1)$
return $\boldsymbol{r}^1$

Amortized SecAnd_1^d cost The SecBernoulli(p/q) gadget performs an average of $(k-1)(1+2^k/q)$ calls to SecAnd_1^d over 1 bit, where $k = \lceil \log_2(q) \rceil$ ($\implies 2^k/q < 2$). This cost accounts for the expected $2^k/q$ iterations of the rejection loop and the final comparison, each requiring a SecLeq gadget against a public constant. When fully bitsliced over a w-bit architecture, this amounts to an amortized average cost of $(k-1)(1+2^k/q)/w$ SecAnd_w^d.

Proposition 6. *Algorithm 14 is PINI. Moreover, on input p/q with $\gcd(p,q) = 1$ and $0 < p < q$, it outputs a Boolean sharing of a sample b which follows a Bernoulli distribution of parameter p/q.*

Proof. **Correctness**. After the rejection loop, U is uniformly distributed in $\{0,\dots,q-1\}$. The output r is 1 if $U \in \{0,\dots,p-1\}$ and 0 otherwise. Since there are exactly p such values, $\Pr(r = 1) = \frac{p}{q}$.

Security (PINI). The gadget is a composition of $\mathsf{SecUnif}^d$ and SecLeq^d, both of which are PINI-secure gadgets. Since the PINI property is stable under composition, the integrity of the sharing is maintained. The only potential leakage arises from $\mathsf{SecUnmask}_1^d(\boldsymbol{b}^1)$. However, since U is generated from fresh randomness and q is a public parameter, the bit b is independent of any sensitive internal state. The declassification of b only reveals the number of iterations, which is statistically independent of the final sample r. Thus, Algorithm 14 is PINI. □

F An unmasked version of Maskaglia

In this section, we present in Algorithm 15 an umasked version of Maskaglia which behaves similarly given the same randomness (excluding of course the masking randomness). It is essentially a more concrete version of Algorithm 6 where the Geom function is instantiated and given in Algorithm 16. We introduce the notation $\mathbb{1}_{x=c}$ for the indicator of the conditon $x = c$, i.e. $\mathbb{1}_{x=c} = \mathbb{1}$ if $x = c$ else it equals 0. Since no masking is used, the various function evaluations of SecDLX can be avoided if we can guarantee that Geom and SecGeom output the same value given the same randomness, since all computations can be derived from the output value of Geom (and a random bit for the sign). The implementation of Geom_p in Algorithm 16 behaves as SecGeom, it goes through all bits (from least significant to most significant) of the randomness and sets the output value to the position of the first bit that was 1.

The implementation used for our benches can be found here: https://github.com/shhun/unmaskaglia.

Alg. 15 :: UNMASKAGLIA$_{s,c,\varepsilon}$

Input : Public parameters standard deviation parameter $s = p/q$, center $c = p'/q' \in Q$, precision parameter $\varepsilon \in \mathbb{R}$, Rényi parameter $a > 1$.
Output : Boolean sharing of a discrete Gaussian sample Y.

// Precomputed constants:
// p_Y, p_K, p_U are the bit sizes of the random variables Y, K, U and B_g, K_{sat} the bounds of Y, K derived from (ε, s, c, a), see Section 4.2

while TRUE **do**
  $G \leftarrow \text{Geom}_{B_g}$
  sign $\leftarrow \mathcal{U}(1)$
  $Y = \mathbb{1}_{\text{sign}=0} \cdot (-\lfloor c \rfloor - G) + \mathbb{1}_{\text{sign}=1} \cdot (\lfloor c \rfloor + G + 1)$
  $K \leftarrow \text{Geom}_{K_{\text{sat}}}$
  $g \leftarrow \gcd(2q^2, p^2 q')$
  $K_0 \leftarrow (2pqq')^2 / g^2$
  $L_Y \leftarrow (2q^2 |Yq' - p'| - p^2 q')/g$
  $L_Y^2 = q_Y \cdot K_0 + r_Y$ // As in Equation (3)
  if $K \geqslant q_Y$ **then**
    $U \leftarrow \mathcal{U}(p_U)$
    $T_Y^{p_U} := \lfloor 2^{p_U} (2^{1 - r_Y/K_0} - 1) \rceil$
    if $K > q_Y \vee (k = q_Y \wedge (U \leqslant T_Y^{p_U}))$ **then break**
return Y

Table 3. Cost of randomness for varying numbers of shares N.

N	2	3	4	5	6
Our work ($c = 0$)	129.9	139.7	154.4	174.0	198.5
Our work ($c = 1/2$)	129.0	137.0	149.0	165.0	185.0
[EGRZ26]	98.5	137.5	196.0	274.0	371.5

G Additional Comparison Metrics

Because our comparison metric of simply evaluating the number of non-linear gates is simplifying, we also compare our scheme to the bitsliced variant of [EGRZ26] in terms of total cost of randomness, and number of linear gates.

G.1 Cost of Randomness

The randomness requirements for a bitsliced masked implementation of HAWK with N shares can be calculated as follows:

- Our work ($c = 0$): $125 + 4.9 \cdot \frac{N(N-1)}{2}$
- Our work ($c = 1/2$): $119 + 4.0 \cdot \frac{N(N-1)}{2}$
- [EGRZ26]: $79 + 19.5 \cdot \frac{N(N-1)}{2}$

Table 3 provides explicit numbers for various numbers of shares.

Alg. 16 :: Geom_p

Output : Geometric sample with p-bit precision

$u \leftarrow \mathcal{U}(p)$
out = 0
prev = $\neg u[0]$
for $i = 1$ to $p - 1$ **do**
  b = prev $\wedge$ $u[i]$
  out = out + $\mathbb{1}_{b=1} \cdot i$
  prev = prev $\oplus$ b
return out

Table 4. Cost of MASKAGLIA sub-routines in terms of 1-bit SecXor gates.

Operation	**Cost**
$\mathsf{SecGeom}_k$	$3k$
SecDLX_k (n functions)	$n(2+3k)+k$
SecEq_k	k
SecLeq_k	$3(k-1)+2$

G.2 Number of Linear Gates

In MASKAGLIA, the cost expressed in 1-bit SecXor gates is given in Table 4. These costs should be multiplied by the rejection factor derived in Corollary 1 and subsequently divided by the machine word width.

In contrast, the cost in terms of 1-bit SecXor gates for the bitsliced approach in [EGRZ26] can be lower-bounded by the cost of the SecLeq comparisons. For the sake of simplicity and to establish a conservative bound, we omit the overhead of the other operations. For a discrete Gaussian with support size n, sampled with k bits of precision, the cost of this method is dominated by $\lceil \log_2(n) \rceil$ comparisons, yielding a total of $\lceil \log_2(n) \rceil \times (3(k-1)+2)$ gates.

In summary, for the parameters used in HAWK, our method requires 22 amortized SecXor_{32} operations per discrete Gaussian sample, whereas the approach in [EGRZ26] requires approximately 30.